PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/60

A1

(11) International Publication Number: **WO 99/63462**

(43) International Publication Date: 9 December 1999 (09.12.99)

(21) International Application Number: PCT/US99/12222

(22) International Filing Date: 2 June 1999 (02.06.99)

(30) Priority Data:

60/087,847	3 June 1998 (03.06.98)	US
09/320,394	26 May 1999 (26.05.99)	US

(71) Applicant: PARETOSCOPE, INC. [US/US]; 7716 Crystal Springs Road, Crystal Lake, IL 60012 (US).

(72) Inventors: HALL, Russell, P., III; 205 Redbud Lane, Chapel Hill, NC 27514 (US). POWSNER, Seth, M.; 285 Ridge Road, Hamden, CT 06517 (US). SHOWALTER, Allan, Ray; 7716 Crystal Springs Road, Crystal Lake, IL 60012 (US). SPITZER, Richard, Alan; 4908 Homerdale Avenue, Toledo, OH 43623 (US).

(74) Agent: POWSNER, David, J.; Choate, Hall & Stewart, Exchange Place, 53 State Street, Boston, MA 02109 (US).

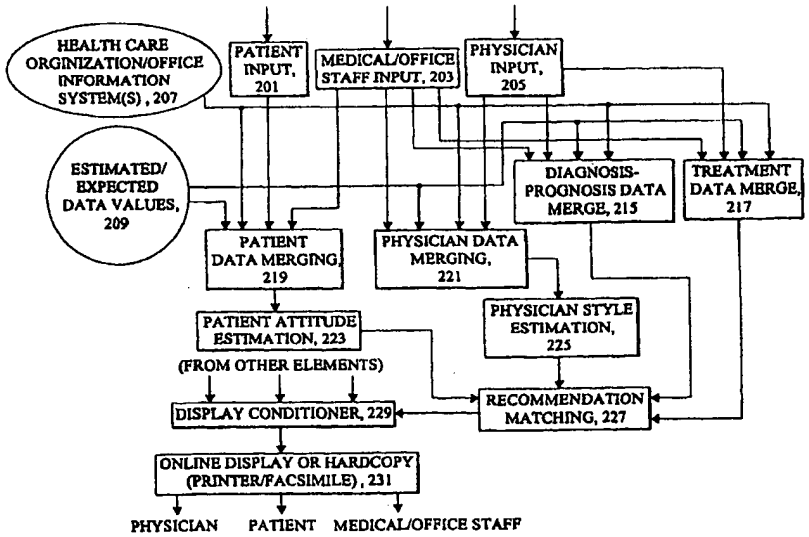
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR PREDICTING AND IMPROVING PATIENT COMPLIANCE WITH MEDICAL TREATMENT



(57) Abstract

Methods and apparatus for predicting and improving patient compliance with a course of medical treatment call for inputting data about the patient (201) and, optionally, about the health care provider (203), the condition diagnosis/prognosis (215), and the treatment recommendation (217). From this data, the patient's attitude or amenability to treatment can be determined, as can the health care provider's attitude and/or ability to induce compliance with a course of treatment. Moreover, methods and apparatus according to the invention can estimate the likelihood of patient compliance with a specific course of treatment or can generate a suggested program of treatment likely to have an increased probability of patient compliance.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND APPARATUS FOR PREDICTING AND IMPROVING PATIENT COMPLIANCE WITH MEDICAL TREATMENT

Reference to Related Applications

This application is a continuation of copending United States Provisional Patent Application No.60/087,847, filed June 3, 1998, the teachings of which application, including the appendices filed therewith, are incorporated by reference herein.

Reference to Appendices

The disclosure of this patent document contains material which is subject to copyright and/or mask work protection. The owner thereof has no objection to facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the U.S. Patent and Trademark Office patent file or records, but otherwise reserves all copyright and/or mask work rights whatsoever.

Background of the Invention

This invention relates to medical therapies and, more particularly, to apparatus and methods for predicting and improving patient compliance with prescribed medical treatments. This invention, which may be used in individual patient treatment as well as group medical care, has application in all fields of medicine, including but not limited to, internal medicine, psychiatry, surgery, obstetrics, pediatrics, and other medical specialties.

In general, medical treatments require patient adherence to a course of action over a period of time. Treatment of hypertension, for example, usually requires taking one or two pills a day and avoiding salty foods. Illnesses as diverse as heart disease, diabetes, advanced arthritis, manic depressive illness, and schizophrenia involve daily

medications. Treatment of low back pain may involve specific exercises over a long period of time. However, even proven cures will fail, if patients do not take them. Cost with no benefit results when patients purchase medications, surgical appliances, or exercise equipment only to leave them in a closet.

Current approaches to improving patient compliance focus on making treatment less onerous for patients and informing and educating patients. Steps to make treatments less onerous include prescribing fewer medications, prescribing medications with fewer side-effects (or side-effects less troubling to a specific patient), prescribing medication according to regimes that reduce the number of times a day the medication must be ingested, and prescribing medication according to regimes that reduce the number of pills that must be ingested each day. Steps to make patients better informed include reviewing medication schedules with patients, reviewing side-effects with patients, reviewing basic medical facts and prognostic information with patients, and providing informational brochures, video tapes or instructional software which allow patients to learn about their illness and their treatments on their own.

Other approaches to improving patient compliance involve automatic reminders for patients. Wrist watches can beep when it is time to take another pill. Somewhat larger timers play back a family member's voice as a reminder. Medication alarm clocks can flash a light to alert patients with hearing loss. Special pill boxes and charts help organize medications. Automated medication dispensers assure the right pills at the right time, and safeguard against overuse of medicine.

More direct, and more expensive approaches to enhance patient compliance involve incentive payments or monitoring. The goal of eradicating tuberculosis has required public health officials, under some circumstances, to employ medical personnel to visit patients daily to directly observe them taking their medication. Pagers with signal-back capability can both deliver a reminder to a patient and accept an acknowledgment from the patient: a form of monitoring without actual visits. Indirect monitoring has been attempted by reviewing pill bottle openings, prescription refills, and physician or hospital visits. These approaches tend to be expensive, e.g.,

requiring electronic pill bottle tops which signal a central office whenever the bottle is opened, or slow, e.g., detecting failure to refill a one month prescription only after a week into the second month of treatment. While these techniques do focus attention on patients whose treatment is not going as planned, they are vulnerable to false negative readings: a pill bottle opening does not necessarily mean that the correct number of pills were taken.

In the foregoing regards, reference may be made to:

R Brian Hynes, M.D., K Ann McKibbon, & Ronak Kanani, "Systematic review of randomized trials of interventions to assist patients to follow prescriptions for medications," Lancet volume 348, pages 383-386, August 10, 1996

Chris Butler, M.B., Ch.B., M.R.C.G.P., Stephen Rollnick, Ph.D., & Nigel Stott, M.D., Ch.B., F.R.C.G.P, F.R.C.P., "The practitioner, the patient and resistance to change: recent ideas on compliance," Canadian Medical Association Journal volume 154, pages 1357-1362, May 1, 1996

Alan Stoudemire, M.D., & Troy Thompson, M.D., "Medication Noncompliance: Systematic Approaches to Evaluation & Intervention," General Hospital Psychiatry, volume 5, pages 223-239, 1983

Nancy Houston Miller, R.N., Martha Hill, Ph.D., R.N., Thomas Kottke, M.D., Ira S. Ockene, M.D., "The Multilevel Compliance Challenge: Recommendations for a Call to Action," Circulation, volume 95, pages 1085-1090, February 18, 1997.

Joyce Cramer & Bert Spilker, Ph.D, M.D. eds, Patient Compliance in Medical Practice and Clinical Trials, Raven Press, New York, 1991

Donald Meichenbaum & Dennis C Turk, Facilitating Treatment Adherence, Plenum Press, New York, 1987

Jack E Fincham, Ph.D., ed, Advancing Prescription Medicine Compliance, Haworth Press, Inc., New York, 1995.

Current approaches to improving medication compliance tend to use one strategy for all patients: individual characteristics are not taken into account. It is up to the treating physician to select among the various approaches. Unfortunately, physicians have proven incapable of accurately predicting which patients will comply with prescribed treatment and which ones will not. Worse, physicians are usually unaware of their inability to predict. Other commonly accepted factors for compliance prediction, such as severity of illness and cost of medication, are now also known to be inaccurate. Nor is compliance accurately predicted by socioeconomic class or cultural background.

The current approaches also assume that patients trust their physicians' recommendations and that patients want to comply with those recommendations (or, at least, believe they would be better off, if they did comply with the recommendations). These approaches further assume that patients react in a reasonably positive manner to their physicians or health care providers. Failure to comply is thus assumed to represent inability, ignorance, or inattention.

An object of this invention, accordingly, is to provide improved methods and apparatus for predicting and improving patient compliance with prescribed medical treatments. More particularly, an object of the invention is to provide such methods and apparatus as take into account characteristics of individual patients in selecting and presenting treatment recommendations in fashions that increase the likelihood those treatments will be followed to completion.

A further object of this invention is to provide such methods and apparatus as take into account the practices and preferences of individual physicians and other

health care providers, in concert with patient characteristics, in selecting and presenting such recommendations.

Yet still another object of the invention is to provide such methods and apparatus as are amenable to automation.

Summary of the Invention

The aforementioned and other objects are attained by the invention, which provides methods and apparatus for predicting and improving patient compliance with a course of medical treatment. Thus, in one aspect, there is provided a method of predicting and/or improving compliance by inputting data about the patient and, optionally, about the health care provider, the condition diagnosis/prognosis, and the treatment recommendation. From this data, the method determines the patient's attitude or amenability to treatment. The method can also determine the health care provider's attitude, or ability to induce compliance with a course of treatment. The method generates, from the input data and corresponding determinations, an estimation of likelihood of patient compliance with a recommended treatment. Alternatively, or in addition, the method can generate a suggested style or program of treatment likely to have an increased probability of patient compliance.

Further aspects of the invention provide methods as described above in which the patient's attitude is determined based on his or her degree of agreement with statements selected on the basis of population survey techniques and, more particularly, for example, on the basis of segmentation process techniques. One aspect of the invention accordingly calls for finding the degree of patient agreement with one or more of the statements:

I'm almost always satisfied with the medical treatment I receive

There are a lot of health care people that you can't trust

Doctors sometimes seem to make decisions before I finish explaining my problems or asking all my questions

I know people who do not exactly follow the doctor's advice and they seem to do just fine

I never know what questions to ask when I go to the doctor with a problem

I try to keep up with current medical information whether I have a problem or not

I've read or heard about new health care treatments that my doctor did not know about

When I get sick or have a serious problem, I do my own research about the illness and treatments

I feel I personally know enough about some medical problems and treatments that I feel comfortable giving advice about them

The costs I pay for health care are reasonable

There is so much information about what is healthy and what is not that it is impossible to make sense of it

I often seek advice from other people

I'm a patient person

I don't like to make long term plans

The quality of many products has gone down

I feel comfortable using the newest technologies

I am very good about planning things in order to keep my appointments

You really do not have much control over what happens to you in life like an illness, it's just a matter of fate or luck

I am pretty good about following healthy eating habits

I wear seatbelts as a driver and as a passenger in a car or truck all the time

Still further aspects of the invention provide apparatus, such as general or special-purpose computers, operating in accord with the methods described above. One such aspect provides, for example, an apparatus that includes an element for inputting patient data, an element for inputting health care provider data (e.g., physician data, medical staff data and health care organization data), diagnosis-prognosis data and treatment recommendation data. Each of these inputs has identifier components, e.g., individual patient and physician and group identifiers, as well as quantitative values, e.g., patient age. The inputs may also have temporal values, e.g., time the quantitative values were measured. An observer identifier may also be provided if the corresponding data is not obtained directly from subject.

A patient signal merging element identifies missing or corrupted inputs for individual patients. This can be based, for example, on group data values. Likewise, a provider signal merging element identifies missing or corrupted input for individual health care providers, again, based on provider group values. A diagnosis-prognosis merging element and a treatment recommendation merging element identify missing or corrupted data in corresponding fashion.

An apparatus according to this aspect of the invention can also include a patient attitude estimating element that analyzes the output from the patient signal merging element. A provider style estimating element likewise analyzes the output

from the provider signal merging element. The attitude and style estimating elements feed their output into a treatment recommendation matcher which also accepts outputs from diagnosis-prognosis and treatment merging elements.

The treatment recommendation matcher generates outputs including, but not limited to, estimations of the probability of patient compliance with the recommended treatment and with the styles of treatment that are likely to have reasonable probabilities of patient compliance.

A display conditioner, which is provided in further aspects of the invention, accepts signals from the treatment recommendation matcher and prepares them for physicians, medical staff, patients, and computerized patient record systems.

According to a further aspect of the invention, an apparatus as described above can include display or printing elements that show display conditioner output, or intermediate signals, in text or graphic form.

In still another aspect, the apparatus can include output elements, e.g., facsimile interface or serial interface or other computer interfaces, to permit different views or recording of display conditioner output or intermediate signals.

In still another aspect, the apparatus can include input elements, e.g., touch screen or keyboard or page scanner or serial interface or other computer interface, to permit either direct or derived information to be used.

In still another aspect, the apparatus can use inputs of a larger device, e.g., an automated instructional system or game, and the apparatus can output to the larger device so that instructional content may be modified by signals generated, e.g., instructional content can be customized.

In still another aspect, the apparatus can generate signals corresponding to group characteristics as it does for individual patient characteristics.

These and other aspects of the invention are evident in the drawings and in the description which follows.

Brief Description of the Drawings

A further understanding of the invention may be attained by reference to the attached drawings, in which:

Figure 1 depicts a computer system used in connection with a preferred practice of the invention;

Figure 2 depicts the overall data flow with a system embodying the invention; and

Figure 3 depicts an exemplary output by an apparatus operating in accord with a preferred practice of the invention.

Detailed Description of the Illustrated Embodiment

Overview

Patients are more likely to comply with medical treatments that are in harmony with their personal styles. An energetic, active person given to athletic hobbies will have difficulty following a treatment regime of rest and warm compresses for a small bone fracture. Such a person may prefer surgical treatment offering quicker return to routine activity, despite surgical risk and cost. On the other hand, a person of philosophical and aesthetic inclination might be quite comfortable with a treatment requiring no surgery and no pills, despite the time required. Even for similar illnesses, different patients may prefer different treatments.

Patients are more likely to comply with medical treatments they believe. Unfortunately, important medical conditions like high blood pressure and high cholesterol are not painful and do not limit activity. Patients must accept such diagnoses based on their physician's interpretation of instrument readings and

laboratory data. Then, they must accept a physician's judgment that their blood pressure or cholesterol is abnormal enough to warrant medication. Finally, they must accept daily medications that have no immediate benefit, though sometimes medications have immediate side-effects. Patients take a lot on faith.

It is difficult convincing patients that they have been correctly diagnosed and that they have been prescribed a personally optimum treatment. It is an uncertain task that is time consuming and costly. Uncertainty can be reduced by determining personal attitudes and styles ahead of time. Then, treatment recommendations can be matched to patient style; and, presentation of medical findings can be matched to patient attitude.

People vary widely in their approach to new information, and from whom they will accept it. Some like news: they seek out news stands and shows and search the World Wide Web. Others are seen with portable tape and compact disk players connected to earphones blocking out even changes in their immediate surroundings. Some people trust information only from established sources with a stake in being factually correct; e.g., Encyclopedia Britannica, major metropolitan newspapers, major network news. Some people prefer the opinions of family, friends and close acquaintances with a stake in their personal well-being. And, even people who share a source may want different details; i.e., all the facts for their own analysis, or, just the bottom line.

Different physicians interact quite differently with their patients. One aspect is their presentation of findings and treatment recommendations. This critical information presentation can be adapted for an individual patient, and more easily than, say, examination technique for a patient's ears, which is proscribed by human anatomy. Trusting patients, who prefer hearing the bottom line, may be more comfortable with just a prescription: detailed explanations may worry them. Skeptical patients might be won over with a detailed explanation and an offer of a separate, follow-up appointment for discussion of treatment options; they might respond to the absence of pressure to commit or comply immediately. Office routine, schedules,

pamphlets and staff are secondary components of physician patient interaction. These can be adapted for specific patient populations if not individual patients.

Predicting and Improving Compliance

The illustrated embodiment predicts and improves patient compliance by 1) gauging patient attitudes and beliefs, 2) gauging physician style of interaction and treatment, then using (1) and (2) along with 3) diagnosis characteristics and 4)treatment characteristics to 5) gauge the match or mismatch between patient and treatment. From result (5), systems according to the invention estimate 6) probability of compliance, 7) subset of treatment characteristics most influencing (6), and 8) style of physician patient interaction most likely to increase compliance (6).

There are no well-accepted techniques for gauging patients' and physicians' attitudes, beliefs and styles within the art of medical compliance. However, the use of demographic data and questionnaires to estimate personality and preferences is well known in other fields. This invention makes novel adaptation of these techniques and combines them with medical information to achieve its purpose.

To avoid bias from commonly accepted factors for compliance prediction, a preferred embodiment of the invention starts with a survey of the population from which patients are drawn. Demographics, opinion survey questions, and medical information are collected from a representative population sample. Cluster analysis yields a few population subgroups whose attitudes, beliefs, behaviors are likely to be similar to each other, and different from people in other clusters.

In market surveys, cluster analysis facilitates segmentation of the market. Medical compliance could be considered a problem in selling medical treatment. However, there are critical differences. Companies start with a product and aim to find customers amidst a large population who will want their product. Treating physicians start with a patient (customer) and aim to find a treatment (product) with which the patient will comply. Products are mass produced; hopefully, customers will come back

for more. Treatments are individualized; hopefully, patients will not need to come back for more. Most important, guided by this invention, a physician can adapt treatment during a patient visit (the moment of sale).

After dividing the population into a suitable set of clusters, each datum from the survey is reanalyzed in terms of its correlation with a person's assigned cluster. The cluster to which a patient belongs is then estimated by collecting information similar to that collected during the survey. Data that does not correlate well with cluster assignment can be skipped. Calculations can be made to compensate for missing data.

In a similar fashion, physician demographics, opinion surveys, and practice patterns are used to determine a small set of physician clusters. For both patients and physicians, cluster analysis derived calculations may be adjusted and may be logically augmented based on experience over time. More complex, symbolic simulations of personal belief systems have been described in the artificial intelligence and psychology literature. Such simulations could be adapted for this invention.

Reanalysis, after cluster determination, also permits match / mismatch relationships to be determined. For each cluster, reviewing group responses makes certain characteristics apparent. For example, if most members of one cluster are highly educated and indicate they like reading news articles about medical discoveries, they can be assumed to prefer detailed explanations about treatments. Similarly, if members of another cluster indicate they are impatient and they have trouble keeping appointments, they are unlikely to comply with treatments based on exercise and diet. If members of a physician cluster indicate they believe their years of clinical training give them knowledge completely unavailable to the lay public, they may need to make a major change in their presentation to a patient whose cluster thinks all important information can be explained in simple terms so they can make their own decisions.

Diagnoses and treatments are characterized along simple dimensions; e.g., severity or risk, visibility, duration, is taking pills involved, is surgery required, etc.

The illustrated embodiment avoids use of a long, exhaustive list of ailments and medications by working with these characterizations. It can handle a new disease and treatment with addition only of a small set of information or specification of an analogous disease and treatment.

Automated Apparatus for Practice of the Invention

Figure 1 illustrates a computer system of the type used in connection with practice of the invention. Computer **101** is connected to data processing peripheral units comprising a flat panel display and touch screen **121**, disk memory **123**, a computer communication network **125**, a pointing device **127**, a facsimile modem **129**, a video monitor **131**, a keyboard **133**, a printer **135**, and a mark sense or page scanner **137**. The disk memory **123** serves as a non-volatile storage element for information accessed by the computer **101**.

The computer communication network **125** serves to allow access to non-volatile storage elements and databases or other peripheral devices among a collection of computers.

The flat panel display and touch screen **121**, the pointing device **127**, the video monitor **131**, and the keyboard **133** provide an interface between the computer **101** and the user. Specifically, the flat panel display and the monitor present a graphic display of signals generated by the computer **101**, while the keyboard **133** and pointing device **133** convert typed messages and positions signals into computer-readable form.

Not all elements are required for practicing the invention, e.g., a flat panel display and touch screen **121** may suffice without a video monitor **131** and a separate pointing device **127**. Likewise, if all input is done directly by users and all output is viewed immediately, a flat panel display and touch screen **121** may suffice without any other input or output devices, e.g., eliminating the need for a printer **135** or scanner **137**.

The illustrated computer **101** includes functional units comprising an I/O controller **104**, a central processor **102**, and a (random access) memory unit **103**. The I/O controller **104** is an interface between the computer **101** and its peripheral units. The central processor **102** serves as the primary source of control, arithmetic, and logic operations within the computer **101**. Further, the memory unit **103** provides volatile, rapid-access storage for the computer **101**, particularly for the central processing unit **102**. The memory unit **103** may also provide non-volatile storage, which, in sufficient quantity, eliminates the need for disk memory **123**.

Software functions **111** are built up from basic operating system & graphical support **118** in a customary manner so that compliance analysis **112** processes data from a variety of sources. Clinical & financial information system **113** data, if not directly available in a specific implementation, can be obtained through database functions **114** using operating system & graphical support **118** making use of computer network communication **125** or modem **129**. Alternatively, such information can obtained from medical staff or directly from patients using accept data **115** and present data **116** user interface functions **117** working through flat panel display & touch screen **121** or through pointing device **127** and video monitor **131** and keyboard **133**.

Those skilled in the art will appreciate that other data input and output routing is possible using the system envisioned in Figure 1, i.e., interactive input and output through the World Wide Web is possible with suitably advanced accept data **115**, present data **116**, and operating system & graphical support **118** coupled with computer network communication **125** or modem **129**. Alternatively, batch input and output is possible in a traditional manner using printer **135** and scanner **137**, or, using clinical financial information system **113** and database functions **114**.

The illustrated hardware used for practice of the invention can be selected from any one of many commercially available programmable digital computers or personnel digital assistants, e.g., preferably, a commercially available Palm Pilot™ running version 2.0 of the Pilot™ operating system, as programmed in accord with the teachings below.

Operation

The structure and operation of the illustrated embodiment is premised on the notion that effective approaches to improving treatment compliance take into account individual patient attitudes and beliefs. Those patients inclined to believe their physicians' judgments and recommendations and whose temperament allows them to follow complicated instructions, for example, might do best with quick, straight-forward prescriptions. Those patients inclined to be skeptical of physicians and medical treatment, on the other hand, might be better served with detailed prognostic information and instructions on watching for developments that they personally believe necessitate treatment. Still others, those patients who prefer a choice of treatment options could receive a selection of possible course of treatment while, those who prefer one "best" recommendation could receive it.

The illustrated embodiment also operates on the assumption that effective treatment takes into account individual physician style and how that interacts with various patient attitudes and beliefs. For example, a dogmatic professional style, and delayed examination due to office overbooking may cause a skeptical and easily insulted patient to *consciously or unconsciously* ignore treatment recommendations as if to *get even.* On the other hand, an accommodating professional style, attentive to individual patient preferences with extra time to talk, may be interpreted as diagnostic uncertainty, or lack of professional stature by a patient who interprets a more authoritarian style as indicative of a technical expert whose time is much in demand.

Ideally, at an individual level, the medical nature and style of a physician's recommendations would be matched to each patient's individual attitudes and beliefs. In public or group medical care, an organization's efforts would be matched to its population's attitudes and inclinations. Extra efforts could be made for patients with serious illness who habitually avoid medical attention. Less effort could be wasted on those who routinely seek medical attention on their own and follow health-care providers' instructions.

Figure 2 depicts a functional interrelationship between elements of a preferred embodiment of the invention, as well as the processing of data by those elements. Patient data merging **219** accepts input, if any, directly from the patient **201** and merges it with patient related data input from medical / office staff input **203**, health care organization / office information systems **207**, and estimated / expected values **209**. Output from patient data merging **219** represents the system's best available characterization of the patient for use in further processing. In a similar fashion physician data merging **221** accepts input from the physician **205**, medical / office staff input **203**, health care organization / office information systems **207**, and estimated / expected values **209.** Output from physician data merging **221** represents the system's best available characterization of the physician, or health care provider, for use in further processing. Diagnosis - prognosis data merging **215** and treatment data merging **217** provide similar data processing for diagnosis-prognosis and treatment respectively.

Complex data signals are passed in this apparatus. These signals include traditional information, e.g., age, sex, weight, blood pressure and laboratory values. These signals include health care utilization information, e.g., patient identifiers, health care provider identifiers, number of office visits, number of prescriptions. These signals include lifestyle and attitudinal information, e.g., hobbies, questionnaire answers, personality and psychological testing results. These signals also include coded diagnoses and treatments, current and past. In a preferred implementation, this information is kept as lists or sets of attribute-value pairs; most of the values are numeric, but non-numeric data and sub-lists are used at times. For instance, treatment information might be as simple as *avoid weight bearing* for a twisted ankle with a *duration - 7 days* and *medication form - none* and *surgery - none*. In another instance, treatment input might be a list of options starting with *antibiotics* for a sexually transmitted disease with a *duration - 10 days* and *medication form - pill* and *surgery-none*, followed by *antibiotics* with a *duration - once* and *medication form - shot in buttocks* and *surgery - none*.

Those skilled in the art will appreciate that estimated or expected data values **209** need not be implemented as an independent entity. Data merging modules **215, 217, 219, & 221** may include constants providing the same information.

Patient attitude estimation **223** analyzes input from patient data merging **219** to characterize the patient's attitude with respect to medical illness and treatment. Physician style estimation **225** analyzes input from physician data merging **221** to characterize the physician or health care provider's style of patient interaction and treatment.

Recommendation matching **227** analyzes the combination of inputs characterizing the overall medical encounter, namely, patient attitude **223**, physician style **225**, diagnosis-prognosis **215** and treatment **217**. Its output estimates the likelihood that the patient will comply with each of the treatments and styles of medical intervention.

Display conditioner **229** accepts input from recommendation matching **227** and converts it into a format, graphical or textual, suitable for the required output **231**. Output might be as simple as a single number proportional to probability that this patient will comply with this physician's recommendation of a single treatment, potentially signaling the need for reconsideration of treatment options. Output may be more complicated to adequately explain to a health care provider what style of treatment might be more likely gain patient compliance.

Those skilled in the art will appreciate that the display conditioner can also accept input from other elements to provide output that monitors the operation of the system. They will also appreciate that conditioner output could be saved on disk memory **123** or stored via database functions **114** when the apparatus is being used to predict compliance for a group of patients or members of a health care organization.

Segmentation

The segmentation process utilized in the illustrated embodiment was developed using well established and accepted methods of population survey techniques. This identified the most relevant content for the survey and the appropriate match to the population being interviewed. For example, asking bus drivers about chicken farming, although using properly framed questions and statistical analysis, is a clear mismatch of content with audience.

A fundamental issue in the research is also to recognize that the process and outcome are not intended to affect any direct behavior or attitudes among people, nor create any artificial experiences. The purpose of the research and the process is to, with the greatest possible accuracy, find ways to describe what already exists; characterizations of people based on their attitudes, behavior and other personal characteristics that accomplishes several things:

reflects their typical and consistent state

provides for discrimination between two or more groups of people, for if all people were identical in their styles, there would be no corresponding need for variety or alternatives in any facet of life

results in a practical method of application in real life activities

describes population groups that are individually large enough to warrant intervention, since each group represents a large number of absolute people, but also avoids a system that defines people in such minute detail that interventions customized to that segment would be economically impractical

utilizes the language, concepts and other portrayals of the population segments that will be most readily identified and accepted because they are, in effect, in their own "language".

The following process was used to develop the segmentation system, including the functional set of 20 questions and related weighting factors to score each person's responses:

Initial sources for segmentation content:

a review of existing literature for ideas, themes, situations, experiences and literal ways of describing them that would be recognized by people;

conducted a series of focus groups with males and females specifically about their experiences and attitudes with a wide variety of health care situations. The objective of the groups was to obtain first hand statements of actual experiences, collect the language used to describe the events and attitudes, and probe for additional underlying issues. This information was then used to frame subsequent survey questions in ways that would replicate what people had related, in the type of language that describes their nature and not those of the researcher.

Segmentation survey:

a national, representative sample of male and female adults was interviewed using an extensive set of questions that was intended to encompass many facets of health care: medical practitioner visits, prescriptions, personal attitudes about health care in general and a person's own health status, and a variety of questions that have been used to describe different activities of people and that may be related to health care, such as participation in sports, nutritional habits, and other lifestyle activities.

The entire data set from the nearly 1700 respondents was analyzed by a series of multi-variate statistical procedures to :

reduce the total number of questions to smaller set of items that statistically derives a set of homogeneous population segments using the reduced set of items and summarizes each construct or theme;

identify items/statements that statistically maximize the difference between the homogeneous population segments;

determine the number of population segments by trying several different solution sizes, such as five groups, six groups, seven groups, etc. The alternatives are tested to accomplish several concurrent objectives:

obtain a final set of segments that is economically practical to work with in terms of the numbers of people involved

provide maximum differentiation between the groups

The original questionnaire contained over 150 items used in the analysis. Since the administration of this number of items in a survey is not practical to administer in any daily medical situation, further statistical analysis was used to reduce the total set to a smaller number, in the case of this study 20 statements, that would replicate the original segmentation and very highly correlate with the allocation of people to specific segments. A basic threshold was set, that the reduced number of questions to be used in typing people must correctly classify (that is, place in the same groups as the full questionnaire) at least 75% of the people. And, any incorrect classification must be in the next closest or adjacent segment for which the person had strong tendencies.

Based on the foregoing, a questionnaire used in the illustrated embodiment obtains the patient's degree of agreement (or disagreement) with the following statements:

I'm almost always satisfied with the medical treatment I receive

There are a lot of health care people that you can't trust

Doctors sometimes seem to make decisions before I finish explaining my problems or asking all my questions

I know people who do not exactly follow the doctor's advice and they seem to do just fine

I never know what questions to ask when I go to the doctor with a problem

I try to keep up with current medical information whether I have a problem or not

I've read or heard about new health care treatments that my doctor did not know about

When I get sick or have a serious problem, I do my own research about the illness and treatments

I feel I personally know enough about some medical problems and treatments that I feel comfortable giving advice about them

The costs I pay for health care are reasonable

There is so much information about what is healthy and what is not that it is impossible to make sense of it

I often seek advice from other people

I'm a patient person

I don't like to make long term plans

The quality of many products has gone down

I feel comfortable using the newest technologies

I am very good about planning things in order to keep my appointments

You really do not have much control over what happens to you in life like an illness, it's just a matter of fate or luck

I am pretty good about following healthy eating habits

I wear seatbelts as a driver and as a passenger in a car or truck all the time

A preferred questionnaire used in posing these questions is provided in the Attachment I. Those skilled in the art will appreciate that more or fewer questions compiled in accord with the foregoing techniques may be used instead of those shown in the attachment. Those skilled in the art will appreciate that these questions may be administered electronically or verbally or on paper and that the responses can be coded numerically.

Outputting Compliance Prediction and Suggestions

Figure 3 illustrates an exemplary individual output generated by the system described above. Particularly, the output reveals a prototypical patent, Mr. John Doe, suffering from hyperlipidemia, who believes he is in good health and who pursues his

own health care information. His questionnaire answers suggest that he is capable of sticking with treatments that require daily adherence over long periods of time; however, they suggest he prefers quick, technically oriented solutions. Under these circumstances, the output warns his physician that it is impossible for anyone to know whether or not Mr. Doe will in fact stick with the treatment recommended. Asking Mr. Doe what he has concluded, after he has had time to check his own sources, may provide the needed information. This output also suggests that exhortations to the effect that Mr. Doe is killing himself by not following treatment recommendations, are unlikely to be heeded. If some other treatment options are going to be considered, a *higher-tech* solution, e.g., cholesterol lowering drugs, may be more acceptable to Mr. Doe despite the expense.

Yet a still further understanding of the invention may be attained by reference to Appendix II, providing a software listing for a preferred embodiment of the invention.

The foregoing describes a preferred apparatus and method for predicting and improving patient compliance achieving the objects set forth above. Those skilled in the art will, of course, appreciate that the illustrated embodiment is exemplary only and that other embodiments incorporating modifications thereto fall with the scope of the invention, of which we claim:

1. A method for any of predicting and improving patient compliance with health care treatment, comprising the steps

A. inputting data relating to a patient and, optionally, data relating to a health care provider, diagnosis-prognosis, and treatment recommendation,

B. determining, from the input data, an attitude of the patient and, optionally, an attitude of the health care provider,

C. generating, from the input data and/or attitude determination, any of an (i) estimation of a likelihood of patient compliance with a recommended treatment and (ii) a style of treatment likely to have an increased probability of patient compliance.

2. A method according to claim 1, comprising the step of determining the patient's degree of agreement with one or more statements selected on the basis of population survey techniques.

3. A method according to claim 2, comprising the step determining the patient's degree of agreement with one or more statements determined on the basis of a segmentation process.

4. A method according to any of claims 1 - 3, comprising the step of determining the patient's degree of agreement or disagreement with one or more of the following statements:

I'm almost always satisfied with the medical treatment I receive;

There are a lot of health care people that you can't trust;

Doctors sometimes seem to make decisions before I finish explaining my problems or asking all my questions;

I know people who do not exactly follow the doctor's advice and they seem to do just fine;

I never know what questions to ask when I go to the doctor with a problem;

I try to keep up with current medical information whether I have a problem or not;

I've read or heard about new health care treatments that my doctor did not know about;

When I get sick or have a serious problem, I do my own research about the illness and treatments;

I feel I personally know enough about some medical problems and treatments that I feel comfortable giving advice about them;

The costs I pay for health care are reasonable;

There is so much information about what is healthy and what is not that it is impossible to make sense of it;

I often seek advice from other people;

I'm a patient person;

I don't like to make long term plans;

The quality of many products has gone down;

I feel comfortable using the newest technologies;

I am very good about planning things in order to keep my appointments;

You really do not have much control over what happens to you in life like an illness, it's just a matter of fate or luck;

I am pretty good about following healthy eating habits; and

I wear seatbelts as a driver and as a passenger in a car or truck all the time.

5. A method according to any of claims 1- 4, comprising the steps of:

A. collecting patient medical data, medical history data, and behavioral and attitudinal data,

B. collecting health care provider data, including that pertaining to diagnosis and prognosis, recommended treatments, style of interaction with patients, treatment statistics, and behavior / attitude,

C. identifying corrupted or missing values among the collected data,

D. estimating attitudes of any of the patient and health care provider based on the collected data,

E. determining a proposed treatment based on numeric and logical analysis of estimated patient and physician attitudes and diagnosis prognosis, and

F. generating an output signal textually or graphically representative of the treatment recommendation with which a specific patient is most likely to comply.

6. A method according to any of claims 1 - 5, comprising obtaining patient and physician data from electronic forms directly or paper forms by suitable scanners or typing.

7. A method according to any of claims 1 - 6, comprising obtaining patient and health care provider data from surrogate sources, such as medical office personnel, heath care organizations, populations statistics.

8. A method according to any of claims 1 - 7, comprising generating text and graphical output on a computer screen or printer as a function of said output signal.

9. A method according to any of claims 1 - 8, comprising displaying treatment inputs ranked in order of probability of patient compliance.

10. A method according to any of claims 1 - 9, comprising displaying treatment characteristics likely to increase probability of patient compliance.

11 A method according to any of claims 1- 10, comprising displaying patient/health care provider interaction styles likely to increase probability of patient compliance.

12. An apparatus operating in accord with the methods of any of claims 1 - 11.

13. A digital data processing apparatus operating in accord with the methods of any of claims 1 - 11.

14. A computer storage media for storing any of instructions and data for causing a digital data processing apparatus to operate in accord with the methods of any of claims 1 - 11.

APPENDIX I

to

METHOD AND APPARATUS FOR PREDICTING AND IMPROVING PATIENT COMPLIANCE WITH MEDICAL TREATMENT

EnrichMap® Health Care and Personal Opinion Questionnaire © ParetoScope, Inc. 1997	Agree Completely 1	2	3	4	Do Not Agree At All 5
I'm almost always satisfied with the medical treatment I receive	_	_	_	_	_
There are a lot of health care people that you can't trust	_	_	_	_	_
Doctors sometimes seem to make decisions before I finish explaining my problems or asking all my questions	_	_	_	_	_
I know people who do not exactly follow the doctor's advice and they seem to do just fine	_	_	_	_	_
I never know what questions to ask when I go to the doctor with a problem	_	_	_	_	_
I try to keep up with current medical information whether I have a problem or not	_	_	_	_	_
I've read or heard about new health care treatments that my doctor did not know about	_	_	_	_	_
When I get sick or have a serious problem, I do my own research about the illness and treatments	_	_	_	_	_
I feel I personally know enough about some medical problems and treatments that I feel comfortable giving advice about them	_	_	_	_	_
The costs I pay for health care are reasonable	_	_	_	_	_
There is so much information about what is healthy and what is not that it is impossible to make sense of it	_	_	_	_	_
I often seek advice from other people	_	_	_	_	_
I'm a patient person	_	_	_	_	_
I don't like to make long term plans	_	_	_	_	_
The quality of many products has gone down	_	_	_	_	_
I feel comfortable using the newest technologies	_	_	_	_	_
I am very good about planning things in order to keep my appointments	_	_	_	_	_
You really do not have much control over what happens to you in life like an illness, it's just a matter of fate or luck	_	_	_	_	_
I am pretty good about following healthy eating habits	_	_	_	_	_
	_	_	_	_	_

I wear seatbelts as a driver and as a passenger in a car or truck all the time

- - - - DS1.394630.4

APPENDIX II

to

METHOD AND APPARATUS FOR PREDICTING AND IMPROVING PATIENT COMPLIANCE WITH MEDICAL TREATMENT

```
//Routines are arranged alphabetically.
//Program execution begins when PilotMain is called by the Pilot Operating
system.

//AppDBClose.c -- Application Database Close; © 1997-8 Seth M Powsner

#include <Pilot.h>
#include "AppDBGlobals.h"
#include "AppDBClose.h"

Err  AppDBClose(void)
{
     Err  error = 0;

     if (appDB)
     {
          //error |= DmResetRecordStates(appDB);     // should be safer but...
          error |= DmCloseDatabase(appDB);
     }
     return error;
}

/*Generic database close adapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
970927    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBClose.h; © 1997-8 Seth M Powsner
extern Err    AppDBClose(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBGlobals.c -- Application Database Globals; © 1997-8 Seth M Powsner
#include <Pilot.h>

DmOpenRef      appDB;
ULong          appDBAppType;
Char           appDBName[dmDBNameLength];
ULong          appDBType;

/*Generic application database globals addapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBGlobals.h; © 1997-8 Seth M Powsner

extern    DmOpenRef              appDB;
extern    ULong                  appDBAppType;
extern    Char                   appDBName[dmDBNameLength];
#define   appDBNameDefault   "AppTestDB"
extern    ULong                  appDBType;
#define   appDBTypeDefault   'Data'

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBInfoInit.c; © 1997-8 Seth M Powsner

#include <Pilot.h>
#include "AppGenericDefs.h"
#include "AppDBGlobals.h"
#include "AppDBInfoInit.h"
#include "AppGlobals.h"
#include "DBOpen.h"
#include "GetResource.h"
```

```
#include "ResourceCommon.h"

static char simpleInfo[] = "App DB info place holder";

Err  AppDBInfoInit(DmOpenRef dbP)
{
     LocalID            appInfoID;
     VoidPtr            appInfoP;
     LocalID            dbID;
     UInt               cardNo;
     VoidHand       h;

     // Get appInfoID given DmOpenRef via dbID and cardNo.
     if (DmOpenDatabaseInfo(dbP, &dbID, NULL, NULL, &cardNo, NULL)
          || DmDatabaseInfo(cardNo, dbID, NULL, NULL, NULL, NULL, NULL,
                                   NULL, NULL, &appInfoID, NULL, NULL,
NULL) )
          return dmErrInvalidParam;
     if (appInfoID == NULL) //create new appInfo if needed
          {
          h = DmNewHandle(dbP, sizeof simpleInfo);
          if (! h) return dmErrMemError;
          appInfoID = MemHandleToLocalID(h);
          DmSetDatabaseInfo(cardNo, dbID, NULL, NULL, NULL, NULL,
                                NULL, NULL, NULL, &appInfoID, NULL, NULL,
NULL);
          }

     // Lock the appInfoID and copy in defaults from our default structure.

     appInfoP = MemLocalIDToLockedPtr(appInfoID, cardNo);
     DmWrite(appInfoP, 0, &simpleInfo, sizeof simpleInfo);

     MemPtrUnlock(appInfoP);
     return OKreturnCode;
}
/*Generic database initialization adapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
970926    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBInfoInit.h; © 1997-8 Seth M Powsner
extern Err    AppDBInfoInit(DmOpenRef dbP);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBOpen.c -- Application Database & Preference Open; © 1997-8 Seth M
Powsner
#include <Pilot.h>
#include "AppDBGlobals.h"
#include "AppDBInfoInit.h"
#include "AppDBOpen.h"
#include "AppGlobals.h"
#include "dbOpen.h"
#include "GetResource.h"
#include "ResourceCommon.h"

Err  AppDBOpen(void)
{
     SystemPreferencesType sysPrefs;

     GetStringRscCopyN(appDBName, primaryDBName, appDBNameDefault,
sizeof(appDBName));
     appDBAppType = GetTypeResource(primaryDBAppType, appType);
     appDBType = GetTypeResource(primaryDBType, appDBTypeDefault);

     PrefGetPreferences(&sysPrefs);
```

```
    if (! (appDB = DBOpen(appDBName, appDBType, appDBAppType,
                                sysPrefs.hideSecretRecords ?
                                    dmModeReadWrite : (dmModeReadWrite |
dmModeShowSecret))))
          return dmErrCantFind;

    return AppDBInfoInit(appDB);
}

/*Generic database open and preference open adapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
970926    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppDBOpen.h  ; © 1997-8 Seth M Powsner

Err  AppDBOpen(void);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppGenericDefs.h -- Application Generic Definitions for convenience    ©
1997 Seth M Powsner
#define OKreturnCode 0
#define  arrayLength(a) (sizeof(a)/sizeof(a[0]))

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppGlobals.c; © 1997-8 Seth M Powsner
#include <Pilot.h>
#include "AppGlobals.h"


ULong          appType;
char           appPreferences[appPrefLen];
int            appVersion;

/*Generic application globals, simplified from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
970926    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppGlobals.h; © 1997-8 Seth M Powsner

extern ULong  appType;
#define            appTypeDefault          'AppT'

#define       appPrefLen              64
extern char        appPreferences[appPrefLen];
#define            appPrefDefault          "rem\0x00blank entry"

extern int         appVersion;
#define       appVersionDefault 100

/* Generic Application Globals
971102 SMP version num * 100
970926 SMP initial
*/

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//ApplicationHandleEvent.c -- generic event application level event handler;
© 1997-8 Seth M Powsner
#include <Pilot.h>
#include "ApplicationHandleEvent.h"
#include "Handlers.h"
```

```
Boolean ApplicationHandleEvent(EventPtr event)
{
    FormPtr   formP;
    Int       formN;
    Boolean   handled = false;

    switch (event->eType)
        {
        case (frmLoadEvent):
            formP = FrmInitForm(formN = event->data.frmLoad.formID);
            FrmSetActiveForm(formP);
            FrmSetEventHandler(formP, FormHandlerTable(formN));
            handled = true;
            break;
        //case ():
        //   break;
        }

    return handled;
}
/*Modification history
980303    SMP       cosmetic
970922    SMP       initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//ApplicationHandleEvent.h  © 1997-8 Seth M Powsner
extern Boolean ApplicationHandleEvent(EventPtr event);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppResourceList.c -- allows correct loading of resource file; © 1997-8
ParetoScope
#include <BuildRules.h>

// Application resources.
char *AppResourceList[] = {
    ":Rsc:" RESOURCE_FILE_PREFIX "Q20.rsrc",
    ""
};

/*Application Resource list module addapted from MemoPadRsc.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modifications
980303    SMP       cosmetic
971115    SMP       reference BuildRules.h and let it set file prefix
971102    SMP       Q20 put in
971012    SMP       simplified version
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//AppResourceList.h  © 1997-8 ParetoScope
#define StringAlert              9000
#define ThreeStringAlert    9001
#define categoryConstants   9950
#define nameEtcForm              1000
#define  resultForm              1100
#define thankYouForm        4100
#define thankYouCode        90
#define answerMultipliers   9
#define lastName            10
#define firstName           20
#define  clientNumber       30
#define resultField              40
#define resultText               90

/* Specific Application Resource ID's
971122    SMP       added result fields & specific forms
971121    SMP       added categoryConstants
971102 SMP old Palm Tutorial definitions removed
971012 SMP initial based on Palm Tutorial 20 MemoPadRsc.h
```

```
*/

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//HandlerHelper.c - convert event info to resource related info for
FormElemHandler;  © 1997-8 Seth M Powsner

#include <Pilot.h>
#include "CommonFormGlobals.h"
#include "CommonFormGlobalFixup.h"

#define ctlTypeListLen 6
ULong ctlTypeList[ctlTypeListLen] = {'tBTN', 'tPBN', 'tCBX', 'tPUT', 'tSLT',
'tREP'};

void CommonFormGlobalFixup(void)
{
     elementRscType = '    ';
     elementID = 0;
     elementPtr = NULL;
     formID = FrmGetActiveFormID();
     formPtr = FrmGetActiveForm();

     switch (event.eType){ //from Event.h enum events,
          //case (nilEvent):
          //case (penDownEvent):
          //case (penUpEvent):
          //case (penMoveEvent):
          //case (keyDownEvent):
          //case (winEnterEvent):
          //case (winExitEvent):
          //                              break;

          case (ctlEnterEvent):
          case (ctlExitEvent):
          case (ctlSelectEvent):
          case (ctlRepeatEvent):
          case (popSelectEvent):              // out of order from enum events,
was after lstExitEvent
                                        elementID =
event.data.ctlEnter.controlID;
                                        elementPtr =
event.data.ctlEnter.pControl;
                                        if (0<= ((ControlPtr)elementPtr)->style
                                             && ((ControlPtr)elementPtr)->style
< ctlTypeListLen)
                                                  elementRscType =
ctlTypeList[((ControlPtr)elementPtr)->style];
                                        break;

          case (lstEnterEvent):
          case (lstSelectEvent):
          case (lstExitEvent):
                                        elementRscType = 'tLST';
                                        elementID = event.data.lstEnter.listID;
                                        elementPtr = event.data.lstEnter.pList;
                                        break;

          case (fldEnterEvent):
          case (fldHeightChangedEvent):
          case (fldChangedEvent):
                                        elementRscType = 'tFLD';
                                        elementID = event.data.fldEnter.fieldID;
                                        elementPtr = event.data.fldEnter.pField;
                                        break;

          case (tblEnterEvent):
          case (tblSelectEvent):
          case (tblExitEvent):                // out of order from enum events,
was after frmCloseEvent
```

```
                                        elementRscType = 'tTBL';
                                        elementID = event.data.tblEnter.tableID;
                                        elementPtr = event.data.tblEnter.pTable;
                                        break;

            case (daySelectEvent): break;
            case (menuEvent):
                                        elementRscType = 'MENU';
                                        elementID = event.data.menu.itemID;
                                        break;
            //case (appStopEvent):
            //case (frmLoadEvent):
            //case (frmOpenEvent):
            //case (frmGotoEvent):
            //case (frmUpdateEvent):
            //case (frmSaveEvent):
            //case (frmCloseEvent):
            //case (tblExitEvent):
            default:                    break;
        }
    }
    /*Modification history
    980303    SMP       cosmetic
    970925    SMP       initial
    */
    /*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

    //CommonFormGlobalFixup.h  © 1997-8 Seth M Powsner
    extern void CommonFormGlobalFixup(void);

    /*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

    //CommonFormGlobals.c -- Commonly used form / event info;  © 1997-8 Seth M
    Powsner

    #include <Pilot.h>
    #include "CommonFormGlobals.h"

    Int           elementID;
    VoidPtr       elementPtr;
    ULong         elementRscType;
    EventType     event;
    Int           formID;
    FormPtr   formPtr;

    /*Adapted from MemoPad.c per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
    Modification history
    980303    SMP       cosmetic
    970925    SMP       initial
    */
    /*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

    //CommonFormGlobals.h  © 1997-8 Seth M Powsner

    extern Int              elementID;
    extern VoidPtr          elementPtr;
    extern ULong        elementRscType;
    extern EventType    event;
    extern Int              formID;
    extern FormPtr          formPtr;

    /*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

    //CustomHandlers.c; © 1997-8 ParetoScope

    #include <Pilot.h>
    #include "AppResourceList.h"
    #include "CommonFormGlobals.h"
    #include "CommonFormGlobalFixup.h"
```

```
#include "CustomHandlers.h"
#include "GetResource.h"
#include "Handlers.h"
#include "ResourceCommon.h"
#include "ScoreCalculation.h"
#include "StoreFieldInput.h"
#include "StoreInput.h"

Boolean NameEtcFormHandler(EventPtr eventP)
{
    Boolean       handled = false;

    event = *eventP;                                    // Copy event data into
CommonFormGlobals
    CommonFormGlobalFixup();

    switch (event.eType){
        case (frmOpenEvent):                            // Open, assume it's
loaded & activated.
            // restore fields if needed
            FrmDrawForm(formPtr);                       // Draw the form
            handled = true;
            break;
        case (ctlSelectEvent):
            if (elementID - formID == nextButtonID) //"start" is a
relabeled "next" button
                ClearInput();                           // clear out checkboxes
            handled = FormElemHandler();      // Leave rest to
GenericControlHandler
            break;
        case (frmCloseEvent):
            SaveFieldInput();
            //handled = true;
            break;
        default:
            handled = FormElemHandler();
        }
    return handled;
}
Boolean ResultFormHandler(EventPtr eventP)
{
    Boolean       handled = false;

    event = *eventP;                                    // Copy event data into
CommonFormGlobals
    CommonFormGlobalFixup();

    switch (event.eType){
        case (frmOpenEvent):                            // Open, assume it's
loaded & activated.
            ScoreCalculation();
            RestoreFieldInput();
            FrmDrawForm(formPtr);                       // Draw the form
            handled = true;
            break;
        case (ctlSelectEvent):
            if (elementID - formID == backButtonID) //"done" is a relabeled
"back" button
                ClearFieldInput();                      // clear out name, etc
            handled = FormElemHandler();      // Leave rest to
GenericControlHandler
            break;
        case (frmCloseEvent):
            // save fields if needed
            //handled = true;
            break;
        default:
            handled = FormElemHandler();
        }
    return handled;
```

```
}

Boolean ThankYouFormHandler(EventPtr eventP)
{
    Boolean       handled = false;
    char *        cp;

    event = *eventP;                                    // Copy event data into
CommonFormGlobals
    CommonFormGlobalFixup();

    switch (event.eType){
        case (frmOpenEvent):                            // Open, assume it's
loaded & activated.
            FrmDrawForm(formPtr);                       // Draw the form
            handled = true;
            break;
        case (keyDownEvent):
            cp = GetStringResource(formID + thankYouCode,"\26");
            if (event.data.keyDown.chr == *cp)
                {
                FrmGotoForm(resultForm);
                handled = true;
                }
            UnlockStringResource(1);
            break;
        default:
            handled = FormElemHandler();
        }
    return handled;
}

/* Modification history
980222    SMP       cosmetic
971122    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//CustomHandlers.h   © 1997-8 ParetoScope

extern Boolean NameEtcFormHandler(EventPtr eventP);
extern Boolean ResultFormHandler(EventPtr eventP);
extern Boolean ThankYouFormHandler(EventPtr eventP);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//DBDefaultCompare.c -- passed to system routines to order database; © 1997-8
Seth M Powsner

#include <Pilot.h>
#include "AppDBGlobals.h"
#include "AppDBInfoInit.h"
#include "DBDefaultCompare.h"

#define   p1smaller     -1
#define   equal         0
#define   p1larger  1

short DBDefaultCompare(void *p1, void *p2, /* only first two parameters are
used*/
                       short other, SortRecordInfoPtr recp1,
                       SortRecordInfoPtr recp2, VoidHand appInfoH)
{
    ULong     len1;
    ULong     len2;
    short     result;

    len1 = MemPtrSize(p1);
    len2 = MemPtrSize(p2);

    if (len1 == 0)
```

```
        result = (len2 == 0) ? equal : plsmaller;
    else if (len2 == 0)
        result = p1larger;
    else
        for (result = equal, len1 = (len1 <= len2) ? len1 : len2;
                        result == equal && len1; ++((CharPtr)p1),
++((CharPtr)p2), --len1)
            if (*(CharPtr)p1 < *(CharPtr)p2)
                result = plsmaller;
            else if (*(CharPtr)p1 > *(CharPtr)p2)
                result = p1larger;
    return result;
}

/*Modification history
980222    SMP       cosmetic changes
971115    SMP       call sequence modified for new SDK
970928    SMP       original
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//DBDefaultCompare.h;© 1997-8 Seth M Powsner
extern DmComparF DBDefaultCompare/* see <DataMgr.h> */;

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

// DBOpen.c; © 1997-8 Seth M Powsner
#include <Pilot.h>                    // all the system toolbox headers
#include "DBOpen.h"

DmOpenRef DBOpen(CharPtr dbName, ULong dbType, ULong appType, UInt mode)
{
    UInt          card;
    LocalID       dbID;
    UInt          dbAttrs;
    DmOpenRef     dbTempP;

    dbTempP = DmOpenDatabaseByTypeCreator(dbType, appType, mode);
    if (!dbTempP && (mode > dmModeReadOnly)) // need to create datebase
        {
        if (!DmCreateDatabase(0 /*card*/, dbName, appType, dbType, false
/*not resource*/))
            {
            dbTempP = DmOpenDatabaseByTypeCreator(dbType, appType, mode);
            if (   !DmOpenDatabaseInfo(dbTempP, &dbID, NULL, NULL, &card,
NULL)
                && !DmDatabaseInfo(0, dbID, NULL, &dbAttrs, NULL, NULL,
NULL,
                                                    NULL, NULL, NULL, NULL,
NULL, NULL)
                )
                {
                dbAttrs |= dmHdrAttrBackup; // assume backup
                if (DmSetDatabaseInfo(0, dbID, NULL, &dbAttrs,
                        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL))
                    dbTempP = NULL; //something's wrong with new db
        } } }
    return (dbTempP);
}
/*Generic routine for manipulating PalmOS database adapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modification history
980222    SMP       cosmetic
970920    SMP       initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

// dbOpen.h; © 1997-8 Seth M Powsner
```

```
extern DmOpenRef DBOpen(CharPtr dbName, ULong dbType, ULong appType, UInt
mode);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//DBWriteOrdered.c-- © 1997-8 Seth M Powsner

#include <Pilot.h>
#include "AppDBGlobals.h"
#include "AppGenericDefs.h"
#include "DBDefaultCompare.h"
#include "dbWriteOrdered.h"

Err DBWriteOrdered(VoidPtr bufferP, ULong bufferLen, Int category)
{
    UInt          attr;
    Err           error;
    Err           errorTemp;
    VoidHand  h = NULL;
    UInt          index = dmMaxRecordIndex;

    // copy buffer into new record at end of database and set category
    error = DmWrite(MemHandleLock(h = DmNewRecord(appDB, &index,
bufferLen)),
                              0, bufferP, bufferLen);
    if ((errorTemp = MemHandleUnlock(h)) != OKreturnCode && !error) error =
errorTemp;
    if ((errorTemp = DmReleaseRecord(appDB, index, true)) != OKreturnCode &&
!error) error = errorTemp;
    if (!error)
        if (!(error = DmRecordInfo(appDB, index, &attr, NULL, NULL)))
        {
            attr = (attr & ~dmRecAttrCategoryMask) | (category &
dmRecAttrCategoryMask);
            if (!(error = DmSetRecordInfo(appDB, index, &attr, NULL)))
                // detach the record and insert it into correct place
                // use same compare routine as for db sorting
                if (!(error = DmDetachRecord(appDB, index, &h)))
                {
                    index = DmFindSortPositionV10(appDB, MemHandleLock(h),
DBDefaultCompare, 0);
                    if (!(error = MemHandleUnlock(h)))
                        error = DmAttachRecord(appDB, &index, h, NULL);
                }
        }
    return error;
}
/*Modification history
980303    SMP       cosmetic
970928    SMP       initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//dbWriteOrdered.h  © 1997-8 Seth M Powsner
extern Err DBWriteOrdered(VoidPtr bufferP, ULong bufferLen, Int category);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//DrawStringInBounds.c-- Draw String in bounds (box) for lists;  © 1997-8
Seth M Powsner
#include <Pilot.h>
#include "drawStringInBounds.h"

void DrawStringInBounds(CharPtr stringP, RectanglePtr bounds)
{
    Int           textLen, width;
    Boolean       fits;
    FontID        currFont;
```

```
    currFont = FntSetFont(stdFont);  // save and set font (a Pilot
convention)

    width = bounds->extent.x - 2;         // determine length of text that
will fit
    textLen = StrLen(stringP);
    FntCharsInWidth(stringP, &width, &textLen, &fits);

    WinDrawChars(stringP, textLen, bounds->topLeft.x, bounds->topLeft.y);
//draw it
    FntSetFont(currFont);                 // restore the font
}
/*Generic Application Resource list module addapted from MemoPad.c
per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
Modifications
980303    SMP       cosmetic
971012    SMP       simplified version
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//DrawStringInBounds.h  © 1997-8 Seth M Powsner
extern void DrawStringInBounds(CharPtr stringP, RectanglePtr bounds);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//EventLoop.c -- get and dispatch events 'till stopped;  © 1997-8 Seth M
Powsner
#include <Pilot.h>
#include "ApplicationHandleEvent.h"
#include "EventLoop.h"

void EventLoop(void)
{
    Word          error;
    EventType     event;

    do   {
         EvtGetEvent(&event, evtWaitForever);
         if (SysHandleEvent(&event)) continue;
         if (MenuHandleEvent(0, &event, &error)) continue;
         if (ApplicationHandleEvent(&event)) continue;
             FrmDispatchEvent(&event);
         }
    while (event.eType != appStopEvent);
}

/*Generic application main routine adapted from MemoPad.c
   per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
980220 SMP cosmetic changes
*/

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//EventLoop.h    © 1997-8 Seth M Powsner
extern void EventLoop(void);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GenericControlHandler.c-- Generic Control event Handler; © 1997-8 Seth M
Powsner

#include <Pilot.h>
#include "CommonFormGlobals.h"
#include "GenericControlHandler.h"
#include "GetResource.h"
#include "ResourceCommon.h"
#include "TestCode.h"

static Boolean GotoFormSafely(int formID);
```

```
Boolean GenericControlHandler(void)
{
     int       effectiveID;
     int       formMultiple = GetIntResource(normalFormMultiple,
formMultipleDefault);
     Boolean handled = false;

     effectiveID = elementID >= formID ? elementID - formID : elementID %
formMultiple;

     switch (event.eType){
          case (ctlSelectEvent):
               switch (effectiveID){
                    case(nextButtonID):
                         {
                         GotoFormSafely(GetIntResource(formID + nextFormSpec,
formID + formMultiple));
                         handled = true;
                         break;
                         }
                    case(backButtonID):
                         {
                         GotoFormSafely(GetIntResource(formID + priorFormSpec,
formID - formMultiple));
                         handled = true;
                         break;
                         }
                    case(restartButtonID):
                         {
                         GotoFormSafely(GetIntResource(firstFormRscID,
firstFormDefault));
                         handled = true;
                         break;
                         }
                    default:
                         {
                         GotoFormSafely(GetIntResource(formID + nextFormSpec,
formID + formMultiple));
                         handled = true;
                         }
               }
               break; // ctlSelectEvent case done
          case (ctlRepeatEvent):
               break;
          case (keyDownEvent):
               switch (event.data.keyDown.chr){
                    case (pageUpChr):
                         {
                         TestCode(formID);
                         //if (!Testcode(form))
                         //    pageUpKeyProcessing();
                         handled = true;
                         break;
                         }
                    case (pageDownChr):
                         {
                         //    pageDownKeyProcessing();
                         //handled = true;
                         break;
                         }
                    default:
                         {
                         //handled = true;
                         break;
                         }
               }
          default: ;
          }
     return handled;
}
```

```
Boolean GotoFormSafely(int formID)
{
    VoidHand tempH;

    if (firstFormDefault <= formID
            && formID <= GetIntResource(lastFormRscID, lastFormDefault)
            &&   NULL != (tempH = DmGet1Resource('tFRM', formID)))
        {
        DmReleaseResource(tempH);
        FrmGotoForm(formID);
        }
    return NULL != tempH;
}
/* control handler to interpret any form's controls based on ResourceCommon.h
980303    SMP       cosmetic
971116    SMP       breakout debugging code and database code
971102    SMP       implement some of generic functions
970925    SMP       initial
*/

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GenericControlHandler.h  © 1997-8 Seth M Powsner
extern Boolean GenericControlHandler(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GenericFormHandler.c; © 1997-8 Seth M Powsner

#include <Pilot.h>
#include "CommonFormGlobals.h"
#include "CommonFormGlobalFixup.h"
#include "GenericFormHandler.h"
#include "Handlers.h"

#include "StoreInput.h" //used in frmClose for checkbox storage
#include "GetObjectPtr.h"

Boolean GenericFormHandler(EventPtr eventP)
{
    Boolean     handled = false;

    int         cID; //used for checkbox save and restore
    ControlPtr  ctlP;

    event = *eventP;                                    // Copy event data into
CommonFormGlobals
    CommonFormGlobalFixup();

    switch (event.eType){
        case (frmOpenEvent):                            // Open, assume it's
loaded & activated.
                                                            // Initialize the form
if needed
            if (2000 <= formID && formID <= 3900  // Not really generic
                && NULL != (cID = RestoreInput(formID))
                && NULL != (ctlP = GetObjectPtr(cID)))
                        CtlSetValue(ctlP, 1);
            FrmDrawForm(formPtr);                       // Draw the form
            handled = true;
            break;
        case (frmCloseEvent):                           // Not really generic,
belongs in question form handler
            if (2000 <= formID && formID <= 3900)
            {
                for (cID = formID + 10; cID <= formID + 50; cID += 10)
//save checkboxes
                    if (CtlGetValue(ctlP = GetObjectPtr(cID)))
                        break;
                if (cID <= formID + 50)
```

```
                    SaveInput(formID, cID);
                }
                break;
            default:
                handled = FormElemHandler();
            }
    return handled;
}
/* Modification history
980222    SMP       cosmetic
971123    SMP       Temporarily added checkbox save / restore code
970925    SMP       Initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GenericFormHandler.h  © 1997-8 Seth M Powsner
extern Boolean GenericFormHandler(EventPtr event);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

// GetObjectPtr.c-- Get (lookup) Object's Pointer based on integer ID;  ©
1997-8 Seth M Powsner
#include <Pilot.h>
#include "GetObjectPtr.h"

VoidPtr GetObjectPtr(Word objectID)
{
    FormPtr frm;
    Word     objIndex;

    if ((objIndex = FrmGetNumberOfObjects(frm = FrmGetActiveForm())) != 0)
        do
            if (FrmGetObjectId(frm, --objIndex) == objectID)
                return (FrmGetObjectPtr(frm, objIndex));
        while (objIndex != 0);
    return(NULL);
}
/*Generic routine adapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
Modification history
980303    SMP       cosmetic
970907    SMP       modified from tutorial to make it safe for non-existent IDs
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GetObjectPtr.h  © 1997-8 Seth M Powsne
extern VoidPtr GetObjectPtr(Word objectID);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GetResource.c-- Get (string and other) Resource(s) in coverted form; ©
1997-8 Seth M Powsner
#include <Pilot.h>
#include "GetResource.h"
#include "StrAToType.h"
#include "StrCopyN.h"

#define   tempHLen  8
static    VoidHand  tempH[tempHLen];
static    Int           tempLast = tempHLen;

Int GetIntResource(Int RscID, Int notFoundReturn)
{
    Int  t;
    CharPtr p;

    p = GetStringResource(RscID, NULL);
    t = p ? StrAToI(p) : notFoundReturn;
    UnlockStringResource(1);
    return t;
}
```

```
CharPtr GetStringResource(Int RscID, CharPtr notFoundReturn)
{
     if (--tempLast < 0) return notFoundReturn;
     if (!(tempH[tempLast] = DmGetlResource(strRsc, RscID))) return
notFoundReturn;
     return MemHandleLock(tempH[tempLast]);
}
CharPtr GetStringRscCopyN(CharPtr dst, Int RscID, CharPtr notFoundReturn,
ULong size)
{
     StrCopyN(dst, GetStringResource(RscID, notFoundReturn), size);
     UnlockStringResource(1);
     return dst;
}
ULong GetTypeResource(Int RscID, ULong notFoundReturn)
{
     ULong     type;
     CharPtr p;

     p = GetStringResource(RscID, NULL);
     type = p ? StrAToType(p) : notFoundReturn;
     UnlockStringResource(1);
     return type;
}
void UnlockStringResource(Int unlockCnt)
{
     if (unlockCnt > 0)
     {
          if (tempLast < 0)
          {
               if (unlockCnt + tempLast <= 0)
               {
                    tempLast += unlockCnt;
                    unlockCnt = 0;
               }
               else
               {
                    unlockCnt += tempLast;
                    tempLast = 0;
               }
          }
          if (unlockCnt + tempLast > tempHLen) unlockCnt = tempHLen -
tempLast;
          for ( ; unlockCnt > 0 ; ++tempLast, --unlockCnt)
               if(tempH[tempLast])
               {
                    ErrNonFatalDisplayIf(
                         MemHandleUnlock(tempH[tempLast]) ||
DmReleaseResource(tempH[tempLast]),
                         "UnlkStrR failed");
                    tempH[tempLast] = NULL;
               }
     }
}

/*Modification history
980304    SMP       cosmetic
970925    SMP       initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GetResource.h  © 1997-8 Seth M Powsner

extern Int GetIntResource(Int RscID, Int notFoundReturn);
extern CharPtr GetStringResource(Int RscID, CharPtr notFoundReturn);
extern CharPtr GetStringRscCopyN(CharPtr dst, Int RscID, CharPtr
notFoundReturn, ULong size);
extern ULong GetTypeResource(Int RscID, ULong notFoundReturn);
extern void UnlockStringResource(Int unlockCnt);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
//GoToRecord.c -- after Find function, may goto record; © 1997-8 Seth M
Powsner
#include <Pilot.h>
#include "AppDBGlobals.h"
#include "GetResource.h"
#include "GoToRecord.h"
#include "ResourceCommon.h"

void GoToRecord (GoToParamsPtr goToParams, Boolean launchingApp)
{
     UInt          attr;
     EventType     event;
     UInt          goToForm;
     Word          recordNum;
     ULong         uniqueID;

     recordNum = goToParams->recordNum; // identify passed rec num
     DmRecordInfo(appDB, recordNum, &attr, &uniqueID, NULL);

     if (!launchingApp)
          {
          FrmCloseAllForms();     // assume app may be running and needs to
clean up
          DmFindRecordByID(appDB, uniqueID, &recordNum); // incase of re-
ordering, use unique ID
          }

     // Send events to load and goto a form selecting the matching text.
     goToForm = GetIntResource(goToFormRscID, firstFormDefault);

     MemSet(&event, sizeof(EventType), 0);

     event.eType = frmLoadEvent;
     event.data.frmLoad.formID = goToForm;

     event.eType = frmGotoEvent;
     event.data.frmGoto.recordNum = recordNum;
     event.data.frmGoto.matchPos = goToParams->matchPos;
     event.data.frmGoto.matchLen = goToParams->searchStrLen;
     event.data.frmGoto.matchFieldNum = goToParams->matchFieldNum;
     event.data.frmGoto.formID = goToForm;
     EvtAddEventToQueue (&event);
}

/*Generic module addapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
Modification history
980221    SMP       cosmetic
971012    SMP       simplified version
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//GoToRecord.h  © 1997-8 Seth M Powsner
extern void GoToRecord(GoToParamsPtr goToParams, Boolean launchingApp);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Handlers.c-- modified for Emap; © 1997-8 ParetoScope
#include <Pilot.h>
#include "AppResourceList.h"
#include "CustomHandlers.h"
#include "CommonFormGlobals.h"
#include "GenericControlHandler.h"
#include "Handlers.h"

#include "GenericFormHandler.h"

FormEventHandlerPtr     FormHandlerTable(Int formN)
{
     switch (formN) {
```

```
//        case (someSpecialForm): return GenericFormHandler; // define in
AppResourceList.h
          case (nameEtcForm):          return NameEtcFormHandler;
          case (resultForm):           return ResultFormHandler;
          case (thankYouForm):    return ThankYouFormHandler;
          default:                     return GenericFormHandler;
          }
}

Boolean FormElemHandler(void)
{
     Boolean        handled = false;

     switch (elementRscType){
          case ('MBAR'):                              //Menu Bar (higher in
EventLoop.c)
               break;
          case ('MENU'):                              //Menu (item)
               switch (elementID){
                    default:  handled = false;
                    }
               break;
          case ('tBTN'):                              // Button
               switch (elementID){
                    default:
                         switch (formID){
                                   default: handled = GenericControlHandler();
                                   }
                    }
               break;
          case ('tCBX'):                              //Check Box
               switch (elementID){
                    default: handled = GenericControlHandler();
                    }
               break;
          case ('tFLD'):                              //Field
               switch (elementID){
                    default: ;
                    }
               break;
          case ('tPBN'):                              //Push Button
               switch (elementID){
                    default: handled = GenericControlHandler();
                    }
               break;
          case ('tPUT'):                              //Popup Trigger
               switch (elementID){
                    default: ;
                    }
               break;
          case ('tREP'):                              //Repeating Button
               switch (elementID){
                    default: handled = GenericControlHandler();
                    }
               break;
          case ('tSLT'):                              //Selector Trigger
               switch (elementID){
                    default: ;
                    }
               break;
          case ('tTBL'):                              //Table
               switch (elementID){
                    default: ;
                    }
          default: handled = GenericControlHandler();

          }
     return handled;
}
/* Form & Form Element handlers that may be customized
```

```
Modifications
980304    SMP       cosmetic
971102    SMP       fixups
970924    SMP       initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Handlers.h  © 1997-8 ParetoScope
extern FormEventHandlerPtr FormHandlerTable(Int formN);
extern Boolean FormElemHandler(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

// Parse.c -- parse (floating numbers); © 1997-8 Seth M Powsner
#include <Pilot.h>
#include <ctype.h>
#include "Parse.h"

double ParseFloat(char **p)
{
     char *c = *p;
     Boolean negative = false;
     double value = 0.0;
     double digitValue = 1.0;

     SkipSpaces(&c);
     if (*c =='-')
          {
          negative = true;
          c++;
          }
     else if (*c == '+')
          c++;
     while ('0' <= *c && *c <= '9') //(isdigit(*c))
          value = value * 10.0 + (float)(*c++ - '0');
     if (*c == '.')
          {
          c++;
          while ('0' <= *c && *c <= '9') //(isdigit(*c))
               {
               digitValue *= 0.1;
               value += (*c++ - '0')*digitValue;
               }
          }
     *p = c;
     return negative ? -value : value;
} // end parseFloat

void SkipSpaces(char **p)
{
     while (**p == ' ' || **p == '\t' || **p == '\n') // (isspace(**p))
          (*p)++;
     return;
}

/* Modification history
980222    SMP       cosmetic
971125    SMP       Eliminate ischar for Pilot linking
971121    SMP       Initial coding, Flp routines would not link correctly so...
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Parse.h  © 1997-8 Seth M Powsner
extern double ParseFloat(char **p);
extern void SkipSpaces(char **p);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//PilotMain.c - called by emulator, returns error code; © 1997 Seth M Powsner
#include <Pilot.h>
```

```
#include "AppGenericDefs.h"
#include "AppDBinfoInit.h"
#include "EventLoop.h"
#include "GoToRecord.h"
#include "InitAppDBInfo.h"
#include "Search.h"
#include "StartApplication.h"
#include "StartForm.h"
#include "StopApplication.h"
#include "SyncNotification.h"

DWord PilotMain(Word cmd, Ptr cmdPBP, Word launchFlags)
{
    switch (cmd) {              // commands are defined in SystemMgr.h

        case (sysAppLaunchCmdNormalLaunch):
            if (StartApplication() == OKreturnCode)
                {
                StartForm(); // start / restart
                EventLoop();
                StopApplication();
                }
            break;

        case (sysAppLaunchCmdFind):
            Search((FindParamsPtr)cmdPBP);
            break;

        case (sysAppLaunchCmdGoTo):
            if (launchFlags & sysAppLaunchFlagNewGlobals)
            {
                if (!StartApplication ())
                    {
                    GoToRecord((GoToParamsPtr) cmdPBP, true);
                    EventLoop();
                    StopApplication();
                    }
            }
            else
                GoToRecord((GoToParamsPtr) cmdPBP, false); //running, just
show record
            break;

        case (sysAppLaunchCmdSaveData):
            FrmSaveAllForms();      //save data, possibly sort db before
possible find
            break;

        case (sysAppLaunchCmdSyncNotify):
            SyncNotification(); //may need to recheck db
            break;

        case (sysAppLaunchCmdInitDatabase): //need to fixup app data in db
(eg, categories)
            AppDBInfoInit(((SysAppLaunchCmdInitDatabaseType*)cmdPBP)->dbP);
            break;
    }
    return 0;
}
/*Generic application main routine adapted from MemoPad.c per CodeWarrior
Targetting Palm OS v.1.2.1 pp.13
Modification history
980220 SMP cosmetic changes
970922 SMP initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

// ResourceCommon.h  © 1997-8 Seth M Powsner

// tSTR resource ID's used to navigate forms and control application
```

```
#define firstFormRscID          9999
#define firstFormDefault        1000
#define  normalFormMultiple          9998          // normal increment
between forms
#define formMultipleDefault          100
#define lastFormRscID           9997
#define lastFormDefault              8900
#define appTypeRsc                   9996
#define appVersionRsc           9995
#define appHeaderStr            9994          // for system find invocations
#define goToFormRscID           9993
#define category00DefaultName   9989
#define category01DefaultName   (category00DefaultName-1)
#define category02DefaultName   (category00DefaultName-2)
#define category03DefaultName   (category00DefaultName-3)
#define category04DefaultName   (category00DefaultName-4)
#define category05DefaultName   (category00DefaultName-5)
#define category06DefaultName   (category00DefaultName-6)
#define category07DefaultName   (category00DefaultName-7)
#define category08DefaultName   (category00DefaultName-8)
#define category09DefaultName   (category00DefaultName-9)
#define category10DefaultName   (category00DefaultName-10)
#define category11DefaultName   (category00DefaultName-11)
#define category12DefaultName   (category00DefaultName-12)
#define category13DefaultName   (category00DefaultName-13)
#define category14DefaultName   (category00DefaultName-14)
#define category15DefaultName   (category00DefaultName-15)
#define primaryDBName           9973
#define primaryDBType           (primaryDBName-1)
#define primaryDBAppType        (primaryDBName-2)
#define secondaryDBName              (primaryDBName-3)
#define secondaryDBType              (primaryDBName-4)
#define secondaryDBAppType      (primaryDBName-5)
#define tertiaryDBName          (primaryDBName-6)
#define tertiaryDBType          (primaryDBName-7)
#define tertiaryDBAppType       (primaryDBName-8)

// tSTR resources whose IDs' are relative to each forms'ID
#define answerLabelforForm      0          // answer label instead of formID
as string
#define nextFormSpec            1          // when normal increment not
desired,
                                           // -1 not allowed
#define priorFormSpec           2          // when normal decrement not
desired,
                                           // -1 not allowed, -2 for pop

/* tSTR resources whose IDs' are relative to each form element's ID (or group
ID)
     so individual resource, then group, then form's tSTR's are searched
     group IDs are added to from ID to make an effective resource ID
   form input element resource ID's should go by 10's
   if a element's ID is less than the form ID, then offset if MOD multiple
     group ID is handled as usual
   home form for element / group is used as back up at element & group steps
*/
#define answerValue             0          // string value to use instead
of implicit
#define answerLabel             1          // answer label instead of form
label
#define answerDefault           2
#define answerMaximum           3
#define answerMinimum           4
#define answerJump              5          // 0 no, -1 yes to form
default, FormID
#define mustAnswer              6
#define freshAnswer             7


// form elements with default intpretations if there is no associated tSTR
```

```
#define nextButtonID            1          // IDs offset from formID
#define  backButtonID           2
#define restartButtonID         3
                    xx
/* Resource ID map
     <=    999          System, according to Tutorial 1.2.2
          1000          first form based on CW Constructor
          1010               first element in first form
          1020               second element,etc
          ....
          1098               second fixed item, etc
          1099               first fixed item in first from, working down
          1100          second form
          1200          third from, etc
          9998          tSTR data
          9999          tSTR data for general form processing, works down
     >    9999          System, based on SystemMgr.H

*/

/*Modification history
980304    SMP       cosmetic
970923    SMP       Initial
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//SaveRestoreField.c -- save and restore chars in a field; © 1997-8 Seth M
Powsner
#include <Pilot.h>
#include "GetObjectPtr.h"
#include "SaveRestoreField.h"
#include "StrCopyN.h"

void SaveField(int ObjectID, char *cp, ULong maxFieldChars)
{
     FieldPtr  fldP;

     if (NULL != (fldP = GetObjectPtr(ObjectID)))
          {
          FldCompactText(fldP);
          StrCopyN(cp, FldGetTextPtr(fldP), fldP->textLen > 0 ? maxFieldChars
: 0);
          }
}
void RestoreField(int ObjectID, char *cp)
{
     FieldPtr  fldP;
     VoidHand  tempH;

     if (NULL != (fldP = GetObjectPtr(ObjectID))
          && NULL != (tempH = MemHandleNew(fldP->maxChars)))
          {
          StrCopyN(MemHandleLock(tempH), cp, fldP->maxChars);
          MemHandleUnlock(tempH);
          FldSetTextHandle(fldP, tempH);
          //FldDrawField(fldP); // but not need before form draw
          }
}

/*Modification history
980222    SMP       initial file
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//SaveRestoreField.h  © 1997-8 Seth M Powsner
extern void SaveField(int ObjectID, char *cp, ULong maxFieldChars);
extern void RestoreField(int ObjectID, char *cp);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
//ScoreCalculation.c -- calculate EnrichMap category score;  ®1997-8
Paretoscope
#include <Pilot.h>
#include "AppResourceList.h"
#include "GetResource.h"
#include "Parse.h"
#include "ScoreCalculation.h"
#include "StoreInput.h"

int       likelyCategory;
double    score[categories];

void ScoreCalculation(void)
{
     char *c;
     int formID;
     int elementID;
     int i;

     for (i = 0; i < categories; i++)
          score[i] = 0.0;

     if (NULL != (c = GetStringResource(categoryConstants, NULL)))
          {
          for (i = 0; i < categories; i++)
               score[i] = ParseFloat(&c);
          UnlockStringResource(1);
          }

     for (FirstInput(&formID, &elementID); formID; NextInput(&formID,
&elementID))
          {
          if (NULL != (c = GetStringResource(formID + 10 + answerMultipliers,
""")))
               {
               for (i = 0; i < categories; i++)
                    score[i] += ParseFloat(&c) * ((elementID - formID) / 10);
               UnlockStringResource(1);
               }
          }

     for (i =  likelyCategory = categories - 1; --i >=0;)
          if (score[likelyCategory] < score[i])
               likelyCategory = i;

     return;
}

/*Modification history
980221    SMP       cosmetic
971124    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//ScoreCalculation.h    ®1997-8 Paretoscope
extern    void      ScoreCalculation(void);
#define             categories 6
extern    int       likelyCategory;
extern    double    score[categories];

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Search.c -- find a record, invoked by system Find feature; © 1997-8 Seth M
Powsner
#include <Pilot.h>
#include "AppDBClose.h"
#include "AppDBGlobals.h"
#include "AppDBOpen.h"
#include "ResourceCommon.h"
#include "drawStringInBounds.h"
#include "GetResource.h"
```

```
#include "Search.h"

void Search (FindParamsPtr findParams)
{
     VoidPtr        appDBRecP;
     UInt           cardNo;
     LocalID        dbID;
     Boolean        full;
     CharPtr        header;
     Word           pos;
     RectangleType r;
     VoidHand  recordH;
     UInt           recordNum;

     if (AppDBOpen() || DmOpenDatabaseInfo(appDB, &dbID, 0, 0, &cardNo, 0))
          findParams->more = false; // no point in being called further if DB
does not open
     else
          {
          header = GetStringResource(appHeaderStr, "Application");      //set
heading
          full = FindDrawHeader(findParams, header);
          UnlockStringResource(1);

          for ( recordNum = findParams->recordNum;       //pickup checking
records where told
                NULL != (recordH = DmQueryNextInCategory(appDB, &recordNum,
dmAllCategories))
                        && !full;
                recordNum++)
                   {
                   if (FindStrInStr(appDBRecP = MemHandleLock(recordH),
                                                    findParams->strToFind, &pos))
                        {
                        if (true == (full = FindSaveMatch(findParams,
recordNum, pos, 0, 0, cardNo, dbID)))
                             recordNum--; // need to visit this record again
                        else
                             {
                             // Get the bounds of the region where we will
draw the results.
                             FindGetLineBounds(findParams, &r);
                             // Display the record in the search dialog.  We
move in one pixel
                             // so that when the record is inverted the left
edge is solid.
                             r.topLeft.x++;
                             r.extent.x--;
                             DrawStringInBounds(appDBRecP, &r);
                             // line is used for the record.
                             findParams->lineNumber++;
                             }
                        }
                   MemHandleUnlock(recordH);
                   }
               findParams->more = (NULL != recordH);
          }
     AppDBClose();
}

/*Generic module addapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
Modification history
980221    SMP       changed stopping logic
971012    SMP       simplified version
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Search.h  © 1997-8 Seth M Powsner
extern void Search (FindParamsPtr findParams);
```

```
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//Serial.c -- serial interface test code; © 1997-8 Seth M Powsner
#include <Pilot.h>
//#include <SerialLibMac.h>
#include <SerialMgr.h>

#define serialPortNum 0
#define defaultBaudRate 57600
#define testMsg "Abc Def\r\n"

static Int    openStatus = false;
static UInt serialLibRefNum;
extern void SerialTest(void);

static Err SerialInit(void)
{
     Err       error = false;
     if (!openStatus)
          if (!(error = SysLibFind ("Serial Library", &serialLibRefNum)))
               if (!(error = SerOpen(serialLibRefNum, serialPortNum,
defaultBaudRate)))
                    openStatus = true;
     return error;
}
void SerialTest(void)
{
ErrNonFatalDisplayIf(SerialInit(), "SerialInit trbl.");
ErrNonFatalDisplayIf(SerClearErr (serialLibRefNum), "SerClearErr trbl."); //
no error returned, just clears
#if EMULATION_LEVEL == EMULATION_NONE
ErrNonFatalDisplayIf(SerSendV10(serialLibRefNum, testMsg, sizeof(testMsg))
                              , "SerSend trbl.");
ErrNonFatalDisplayIf(SerSendWait(serialLibRefNum, serDefaultCTSTimeout)
                              , "SerSendWait trbl."); //5*sysTicksPerSecond
#endif
ErrNonFatalDisplayIf(SerClose(serialLibRefNum)
                              , "SerClose trbl."); // serErrNotOpen may not be
error
}

/*Modification history
980221    SMP       cosmetic
971115    SMP       initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StartApplication.c -- initialization, including opening main database; ©
1997-8 Seth M Powsner
#include <Pilot.h>
#include "AppGlobals.h"
#include "AppDBGlobals.h"
#include "AppDBOpen.h"
#include "GetResource.h"
#include "ResourceCommon.h"
#include "StartApplication.h"
#include "StoreFieldInput.h"
#include "StrCopyN.h"

Boolean StartApplication(void)
{
     //SystemPreferencesType sysPrefs;

     //PrefGetPreferences (&sysPrefs);

     appType = GetTypeResource(appTypeRsc, appTypeDefault);
     appVersion = GetIntResource(appVersionRsc, appVersionDefault);
     if (!PrefGetAppPreferencesV10(appType, appVersion, appPreferences,
sizeof (appPreferences)))
     {
```

```
            StrCopyN(appPreferences, appPrefDefault, sizeof(appPreferences));
            PrefSetAppPreferencesV10(appType, appVersion, appPreferences,
sizeof(appPreferences));
        }
        // Clear out name fields
        ClearFieldInput();

        return AppDBOpen();
}

/*Generic routine adapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
980221 SMP cosmetic changes
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StartApplication.h  © 1997-8 Seth M Powsner
extern Boolean StartApplication(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StartForm.c -- go to first form based on resource definitions; © 1997-8
Seth M Powsner
#include <Pilot.h>
#include "GetResource.h"
#include "ResourceCommon.h"
#include "StartForm.h"

void StartForm(void)
{
        FrmGotoForm(GetIntResource(firstFormRscID, firstFormDefault));
}

/*Generic application main routine adapted from MemoPad.c
   per CodeWarrior Targetting Palm OS v.1.2.1 pp.13
980220 SMP cosmetic changes
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StartForm.h                © 1997-8 Seth M Powsner
extern void StartForm(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StopApplication.c -- backout things setup in StartApplication;  © 1997-8
Seth M Powsner
#include <Pilot.h>
#include "AppGlobals.h"
#include "AppDBClose.h"
#include "StopApplication.h"

void StopApplication(void)
{

        FrmCloseAllForms();     // frmCloseEvent to all forms (outside of
EventLoop control)

        // Close database
        ErrNonFatalDisplayIf(AppDBClose(), "Trouble closing DB.");

        // Save preferences & state initialized in StartApplication
        PrefSetAppPreferencesV10(appType, appVersion, appPreferences,
sizeof(appPreferences));

}

/*Generic routine adapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
Modification History
980221    SMP       cosmetic changes
```

```
971115    SMP       V10 call for compatibility under new SDK
970927    SMP       original adapted from Tutorial
*/

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StopApplication.h  © 1997-8 Seth M Powsner
extern void StopApplication(void);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StoreFieldInput.c -- store (and restore) name fields, etc; © 1997-8
ParetoScope
#include <Pilot.h>
#include "AppGenericDefs.h"
#include "AppResourceList.h"
#include "CommonFormGlobals.h"
#include "GetResource.h"
#include "SaveRestoreField.h"
#include "ScoreCalculation.h"
#include "StoreFieldInput.h"

int           fieldList[]        = {lastName, firstName, clientNumber};
const int     maxFieldChars      = 32;
char          fieldString[arrayLength(fieldList)][maxFieldChars];

void ClearFieldInput(void)
{
     int  i;
     for (i = arrayLength(fieldList); --i >= 0; )
          fieldString[i][0] = '\0';
}
void SaveFieldInput(void)
{
     int  i;
     for (i = arrayLength(fieldList); --i >= 0; )
          SaveField(fieldList[i] + formID, fieldString[i], maxFieldChars);
}
void RestoreFieldInput(void)
{
     int  i;
     for (i = arrayLength(fieldList); --i >= 0; )
          RestoreField(fieldList[i] + resultForm, fieldString[i]);
     RestoreField(resultField + resultForm,
                  GetStringResource(likelyCategory + resultText + formID,
"Error. Please report."));
     UnlockStringResource(1);
}

/* Modification history
980222    SMP       separated out low level Pilot calls, & cosmetic fixes
971125    SMP       Use likelyCategory to set response
971123    SMP       Initial coding
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StoreFieldInput.h © 1997-8 ParetoScope
extern void ClearFieldInput(void);
extern void SaveFieldInput(void);
extern void RestoreFieldInput(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StoreInput.c -- store inputs from form; © 1997-8 Seth M Powsner
#include <Pilot.h>
#include "StoreInput.h"

const tableLen = 29; //should be prime
typedef struct { int formID; int elementID; } formElementPair;
const empty = 0;
```

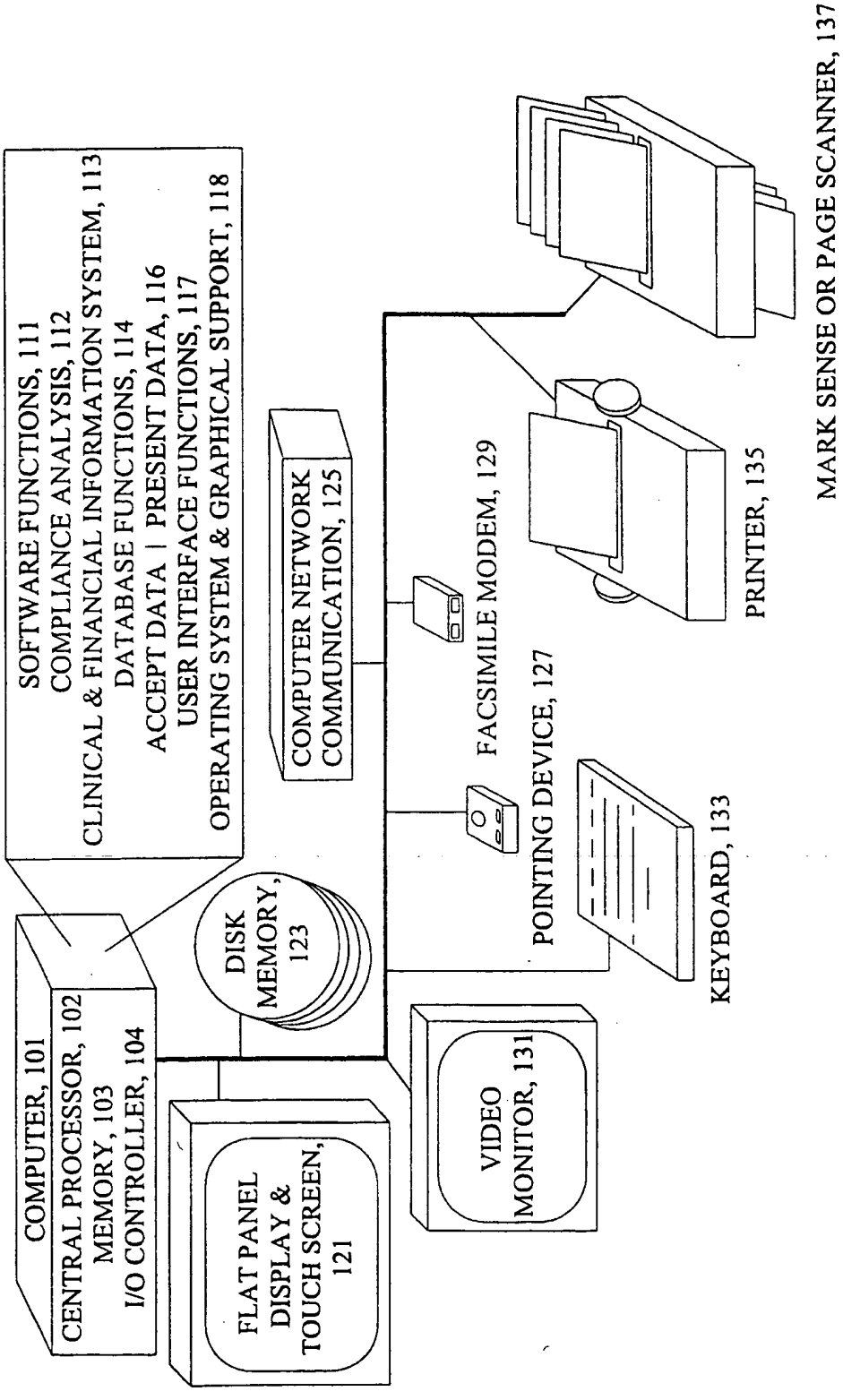
```
static formElementPair table[tableLen];
Boolean tableInitialized = false;

formElementPair *possibleEntry(int formID);

void ClearInput(void)
{
    tableInitialized = false;
}

void SaveInput(int formID, int elementID)
{
    formElementPair *p;

    ErrFatalDisplayIf(NULL == (p = possibleEntry(formID))
                      || (p->formID != formID && p->formID != empty),
"Input table overflow.");
    p->formID = formID;
    p->elementID = elementID;
    return;
}

int RestoreInput(int formID)
{
    formElementPair *p = possibleEntry(formID);

    return p && p->formID == formID ? p->elementID : NULL;
}

formElementPair *possibleEntry(int formID)
{
    int probe, toCheck;

    if (!tableInitialized)
        {
        tableInitialized = true;
        for (probe = tableLen; --probe >=0;)
            table[probe].formID = empty;
        }
    for (probe = formID % tableLen, toCheck = tableLen;
         --toCheck >= 0 && table[probe].formID != empty &&
table[probe].formID != formID;
         probe = (probe + 1) % tableLen)
        ;
    return &table[probe];
}

int nextPair = -1;

void FirstInput(int *formIDp, int *elementIDp)
{
    nextPair = tableLen - 1;
    NextInput(formIDp, elementIDp);
}

void NextInput(int *formIDp, int *elementIDp)
{
    while (nextPair >= 0 && table[nextPair].formID == empty)
        --nextPair;
    if (nextPair >= 0)
        {
        *formIDp = table[nextPair].formID;
        *elementIDp = table[nextPair].elementID;
        --nextPair;
        }
    else
        *formIDp = *elementIDp = empty;
    return;
}
```

```
/* Modification history
980222    SMP       cosmetic
971121    SMP       Initial coding, simple hash design, element ID only
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StoreInput.h  © 1997-8 Seth M Powsner
extern void ClearInput(void);
extern void FirstInput(int *formIDp, int *elementIDp);
extern void NextInput(int *formIDp, int *elementIDp);
extern void SaveInput(int formID, int elementID);
extern int RestoreInput(int formID);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StrAToType.c -- convert ascii to "type" type;  © 1997-8 Seth M Powsner
#include <Pilot.h>
#include "StrAToType.h"

ULong StrAToType(CharPtr str)
{
     Int       n;
     CharPtr p;
     ULong     type = '    ';

     for (n = sizeof(type), p = (CharPtr)&type; --n >= 0 & *str;)
          *p++ = *str++;
     return type;
}
/*Modification history
980222    SMP       cosmetic
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StrAToType.h  © 1997-8 Seth M Powsner
extern ULong StrAToType(CharPtr str);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StrCopyN.c -- string copy with limit;  © 1997-8 Seth M Powsner
#include <Pilot.h>
#include "StrCopyN.h"

CharPtr StrCopyN(CharPtr dst, CharPtr src, ULong n)
{
     if (n != 0)
          while (--n && (*dst++ = *src++) != 0) ;
     if (n == 0) *dst++ = NULL;
     return dst;
}

/*Modification history
980221    SMP       cosmetic
970926    SMP       initial coding (std function, but not on Pilot)
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//StrCopyN.h  © 1997-8 Seth M Powsner
extern CharPtr StrCopyN(CharPtr dst, CharPtr src, ULong n);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//SyncNotification.c -- let application know about syncronizations; © 1997-8
Seth M Powsner
#include <Pilot.h>
#include "SyncNotification.h"

Err SyncNotification(void) //dummy routine
{
     return (0);
}
```

```
/*Generic routine adapted from MemoPad.c per CodeWarrior Targetting Palm OS
v.1.2.1 pp.13
Modification history
980221    SMP       cosmetic
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//SyncNotification.h  © 1997-8 Seth M Powsner
extern Err SyncNotification(void);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//TestCode.c-- generic Test Code routine stub; © 1997-8 Seth M Powsner
#include <Pilot.h>
#include "AppResourceList.h"
#include "TestCode.h"

Boolean TestCode(int formID)
{
     //SoundManager.h
     SndPlaySystemSound(sndWarning); //to signal code being tested
     //add code here
     return true;
}
/*Modification history
980303    SMP       cosmetic
971120    SMP       initial use
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//TestCode.h  © 1997-8 Seth M Powsner
Boolean TestCode(int formID);
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//ToStr.c -- various type to string conversions;  © 1997-8 Seth M Powsner
#include <Pilot.h>
#include <NewFloatMgr.h>
#include "ToStr.h"

static char * StrUToA2(char *p, ULong u, Boolean leadingZero);

static char convertBuf[8][12];
static Word convertBufNext = 0;
#define convertBufNextMask 7;

char * FloatToStr(double d)
{
     FlpCompDouble *p = (void *) &d;
     convertBufNext = (convertBufNext + 1) & convertBufNextMask;
     FlpFToA(p->fd, convertBuf[convertBufNext]);
     return convertBuf[convertBufNext];
}

char * IntToStr(/*long*/int i)
{
     convertBufNext = (convertBufNext + 1) & convertBufNextMask;
     return StrIToA(convertBuf[convertBufNext], i);
}
char * ULongToStr(ULong i)
{
     convertBufNext = (convertBufNext + 1) & convertBufNextMask;
     return StrUToA2(convertBuf[convertBufNext], i, false);
}
char * ULongToStrFixed(ULong i)
{
     convertBufNext = (convertBufNext + 1) & convertBufNextMask;
     return StrUToA2(convertBuf[convertBufNext], i, true);
}

static ULong powersOfTen[] = {1000000000, 100000000, 10000000,
```

```
                                        1000000,    100000,    10000,
                                           1000,       100,       10,
                                              0};
char * StrUToA2(char *p, ULong u, Boolean leadingZero)
{
     char      *dp = p;
     int       i;
     ULong     *tenp;

     for (tenp = &powersOfTen[0]; *tenp; tenp++)
          {
          i = (int) (u/(*tenp));
          leadingZero |= (i != 0);
          if (leadingZero)
               *dp++ = ("0123456789")[i];
          u -= i * (*tenp);
          }
     *dp++ = ("0123456789")[u];
     *dp++ = 'u';
     *dp = NULL;
     return p;
}

/*Modification history
980222    SMP       cosmetic
971115  SMP         added StrUToA & StrUtoAFixed
971113    SMP       renamed ToStr.c, ULongToStr added;
970921    SMP       separated out
*/
/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

//ToStr.h  © 1997-8 Seth M Powsner
extern char * FloatToStr(double d);
extern char * IntToStr(/*long*/int i);
extern char * ULongToStr(ULong i);
extern char * ULongToStrFixed(ULong i);

/*EOF - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

FIG. 1


COMPUTER, 101
CENTRAL PROCESSOR, 102
MEMORY, 103
I/O CONTROLLER, 104
SOFTWARE FUNCTIONS, 111
COMPLIANCE ANALYSIS, 112
CLINICAL & FINANCIAL INFORMATION SYSTEM, 113
DATABASE FUNCTIONS, 114
ACCEPT DATA | PRESENT DATA, 116
USER INTERFACE FUNCTIONS, 117
OPERATING SYSTEM & GRAPHICAL SUPPORT, 118
FLAT PANEL DISPLAY & TOUCH SCREEN, 121
DISK MEMORY, 123
COMPUTER NETWORK COMMUNICATION, 125
VIDEO MONITOR, 131
POINTING DEVICE, 127
FACSIMILE MODEM, 129
KEYBOARD, 133
PRINTER, 135
MARK SENSE OR PAGE SCANNER, 137

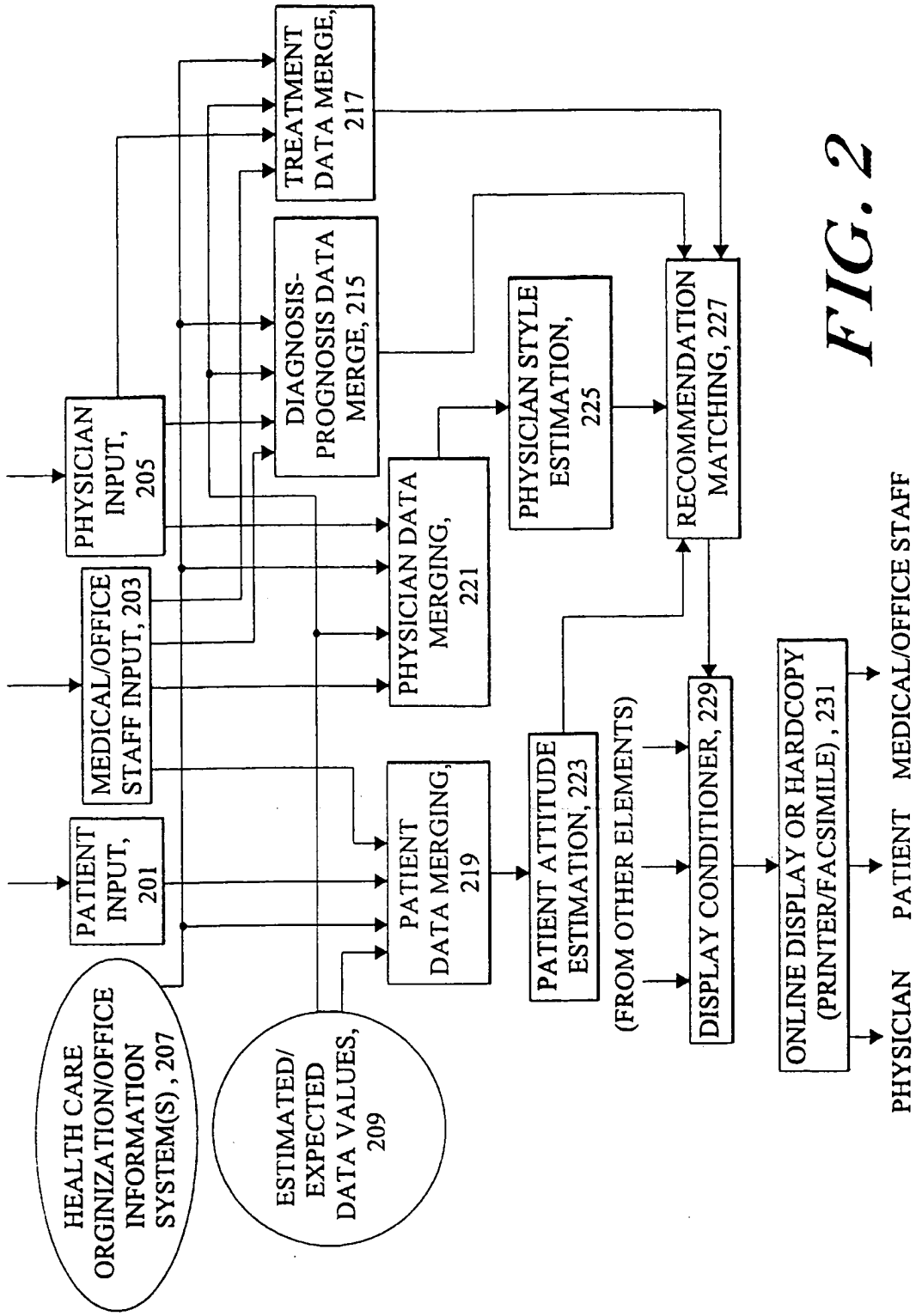

HEALTH CARE ORGINIZATION/OFFICE INFORMATION SYSTEM(S), 207
ESTIMATED/ EXPECTED DATA VALUES, 209
PATIENT INPUT, 201
MEDICAL/OFFICE STAFF INPUT, 203
PHYSICIAN INPUT, 205
PATIENT DATA MERGING, 219
PHYSICIAN DATA MERGING, 221
DIAGNOSIS-PROGNOSIS DATA MERGE, 215
TREATMENT DATA MERGE, 217
PATIENT ATTITUDE ESTIMATION, 223
PHYSICIAN STYLE ESTIMATION, 225
(FROM OTHER ELEMENTS)
RECOMMENDATION MATCHING, 227
DISPLAY CONDITIONER, 229
ONLINE DISPLAY OR HARDCOPY (PRINTER/FACSIMILE), 231
PHYSICIAN
PATIENT
MEDICAL/OFFICE STAFF

FIG. 2

Mr. John Doe	
Is a	middle aged man
Who believes he is	in good health.
Medical attitudes are similar to an individual who is an	informed health care knowledge seeker.
Most likely to be convinced of the diagnosis	hyperlipidemia
with possibly	poor long term prognosis
by	collegial, factual presentations of the factors leading to this diagnosis. Scientific articles describing the desease and it's outcomes may be helpful; but, he will likely pursue the subject on his own.
Questionnaire answers suggest he is	capable of sticking with a plan
for	change of diet and increased exercise, but only once he is fully convinced.
For this particular plan at this particular time,	probability of compliance is indeterminate from information provided.
Consider	follow-up contact in two to four weeks to solicit his opinion about the diagnosis and treatment recommended.
He tends to	prefer quick fix treatment options based on his inclination towards new technology and his prior surgery.

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/12222

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/60
US CL :705/3, 2

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/3, 2, 1, 10; 600/300; 128/906, 920, 921, 923, 924

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 5,642,731 A (KEHR) 01 July 1997, see figure 18; col. 4, lines 20-32 and 40-53; col. 7, lines 29-60; col. 13, lines 52-58; col. 18, lines 16-25; col. 19, lines 22-35; col. 21, lines 33-67.	1-3 ---- 4
Y	US 5,572,421 A (ALTMAN ET AL.) 05 November 1996, see col. 1,1 ines 20-30; col. 2, lines 23-33; col. 4, lines 1-5 and 42-45; col. 14, lines 10-26; col. 16, lines 16-26; appendix II.	4
A,P	US 5,908,383 A (BRYNJESTAD) 01 June 1999, see abstract; figure 5; col. 2, lines 25-46.	2

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 11 SEPTEMBER 1999	Date of mailing of the international search report 28 SEP 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer TODD VOELTZ Joni Hill Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/12222

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 5-14
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/12222

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

APS, Dialog
patient compliance, patient survey, patient attitude, treatment style, diagnosis, patient question, population survey, consent, noncompliance, questionnaire